1991-09

A framework for classifying and resolving semantic teterogeneity in object-oriented databases

Bourque, Michael T.

Monterey, California. Naval Postgraduate School

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (*If applicable*) 55	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000	7b. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*)	10. SOURCE OF FUNDING NUMBERS Program Element No. / Project No. / Task No. / Work Unit Accession Number

11. TITLE (*Include Security Classification*)
A FRAMEWORK FOR CLASSIFYING AND RESOLVING SEMANTIC HETEROGENEITY IN OBJECT-ORIENTED DATABASES (UNCLASSIFIED)

12. PERSONAL AUTHOR(S) Bourque, Michael T.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*year, month, day*)	15. PAGE COUNT
Master's Thesis	From To	September, 1992	109

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES			18. SUBJECT TERMS (*continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUBGROUP	Databases, Object-Oriented Analysis, Semantic Heterogeneity

19. ABSTRACT (*continue on reverse if necessary and identify by block number*)

During the past three decades, many organizations have seen a dramatic proliferation of a variety of information systems. Organizations soon discovered the need to access and share data across these different information systems. Under current technology, this integration is usually not possible due to the heterogeneity of information systems. One level of heterogeneity is that of semantics. The objective of this thesis is to build a framework for enumerating, classifying , and resolving the types of semantic heterogeneity that could exist in an object-oriented database model. The framework covers both schema and data content conflicts. The schema conflicts are classified broadly by the level at which they occur. The primary data conflicts covered include inconsistencies and different representations for the same data.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS REPORT ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Magdi Kamel	22b. TELEPHONE (*Include Area code*) (408) 646-2494 — 22c. OFFICE SYMBOL AS/KA

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

Approved for public release; distribution is unlimited.

A Framework for Classifying and Resolving Semantic Heterogeneity in Object-Oriented Databases

by

Michael T. Bourque
Lieutenant, United States Navy
B.A., University of Rochester, 1983

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September, 1992

ABSTRACT

During the past three decades, many organizations have seen a dramatic proliferation of a variety of information systems. Organizations soon discovered the need to access and share data across these different information systems. Under current technology, this integration is usually not possible due to the heterogeneity of information systems. One level of heterogeneity is that of semantics. The objective of this thesis is to build a framework for enumerating, classifying, and resolving the types of semantic heterogeneity that could exist in an object-oriented database model. The framework covers both schema and data content conflicts. The schema conflicts are classified broadly by the level at which they occur. The primary data conflicts covered include inconsistencies and different representations for the same data.

c.1

TABLE OF CONTENTS

I. INTRODUCTION

A. BACKGROUND

During the past three decades, many organizations have seen a dramatic proliferation of a variety of information systems. While these information systems are useful in supporting their different activities, organizations soon discovered the need to access and share data across these different information systems.

Under current technology, this integration is usually not possible due to the heterogeneity of information systems. This heterogeneity exists at three basic levels (Bertino, 1989). The first is the information system level. Data is managed by a variety of information systems based on different data models and languages. The second level of heterogeneity is that of semantics. Since different information systems have been designed independently, semantic conflicts are likely to be present. This includes both schema (e.g., name, type conflicts) and data (e.g., inconsistencies) conflicts. Finally, the third level of heterogeneity is that of hardware, operating systems, and communications.

Several approaches have been proposed to address the issues of integrating heterogeneous information systems (Sheth, 1990, pp.183-236). A common theme of these approaches

is the need for a semantically rich integrating model to represent, resolve the conflicts of, and integrate the different component information systems. In this thesis the issues of identifying and resolving semantic conflicts by using a generic object-oriented data model as the integrating model are examined.

B. OBJECTIVES

The objective of this thesis is to build a framework for enumerating, classifying, and resolving the types of semantic heterogeneity that could exist in an object-oriented database model. The framework will cover both schema and data content conflicts. The schema conflicts are classified broadly by the level at which they occur. These levels are: object level conflicts, attribute level conflicts, object-attribute level conflicts, and object method conflicts. The primary data conflicts covered include inconsistencies and different representations for the same data.

To accomplish the objective, a real world database scenario is presented, a generic object-oriented model is presented, and the conflict framework is proposed.

C. RESEARCH QUESTIONS

1. Can a workable framework for classifying and enumerating schema and data heterogeneity conflicts in an object-oriented database models be developed?

2. Can proposed solution guidelines to identified schema and data heterogeneity conflicts in object-oriented database models be developed?

D. SCOPE AND LIMITATIONS

This thesis will briefly describe the three levels of heterogeneity. It will then focus on building a framework for enumerating and classifying schema and data conflict in an object-oriented database model and propose a guideline for conflict resolutions. A similar framework is presented for use with a relational model. Information systems level, hardware, operating systems, and communications heterogeneity will not be addressed in this thesis.

E. METHODOLOGY

This research started with a literature review of pertinent topics including: object-oriented models, object-oriented databases, federated database systems, multidatabase systems, schematic and data heterogeneity issues, information systems proliferation issues, and specific Department of Defence and Department of the Navy information systems proliferation problems. The second step was to identify the generic object-oriented model used in this research. The third step was to identify a useable real world database scenario to use for research. The fourth step was the development of the proposed framework. The final step was the

development of the guidelines for resolving the identified conflicts.

F. ORGANIZATION OF THESIS

The organization of the remainder of the thesis is as follows. Chapter II explains the background of the issue, presents a rationale of why the problems related to heterogeneity evolved, and explains the different types of heterogeneity in information systems. Chapter III overviews a framework for classifying schematic and data conflicts in a relational model. Chapter IV presents the main characteristics of the object-oriented model used in this research. Chapter V presents the database scenario used in this research. Chapter VI develops a framework for classifying the schematic and data conflicts of the object-oriented model presented in Chapter IV and uses the database scenario in Chapter V to illustrate conflict examples. Chapter VII presents guidelines for resolving the conflicts identified in Chapter VI. Chapter VIII concludes the paper with a summary and provides directions for future research.

II. BACKGROUND ON HETEROGENEOUS DATABASES

A. BACKGROUND

Today, from administrative to operational commands, the use of computers to solve data manipulation problems is very common. This fact had its foundations with the widespread use of mainframe computers in the 1960s. In the military, these early computers were mostly associated with data and research centers. Initially, programs were developed for specialized applications that relied on large amounts of data typically stored on disks. The data was stored in flat file systems and uniquely addressed by the programs developed to use it. It was apparent that a large amount of information was common to different applications and that there was a need to share access to data (Parsaye, 1989, pp.36).

As technology was introduced to organizations, the type of administrative control within the organization dictated how new technology was exploited. The DOD had a slack environment when database technology was introduced. Each branch of the service was allowed to operate independently with little or no guidance. Within the services, major commands also acted independently. Though this lack of coordination led to many duplicate efforts and a lack of standardization, this approach

had some benefits. To quote Richard Nolan (Nolan's stages of growth);

> "The balance between control and slack is important in developing appropriate management approaches for each stage of an organizational learning. For example, an imbalance of high control and low slack in the earlier stages can impede the use of information technology in the organization; conversely, an imbalance of low control and high slack in the latter stages can lead to explosive data processing budget increases and inefficient systems." (Nolan, 1979, pp.117)

In the March-April 1979 Harvard Business Review Richard Nolan wrote an article "Managing the Crisis in Data Processing." In this article Nolan proposed six stages of growth. The first is the initiation stage where new technology is first introduced. The second stage is the contagion stage where proliferation of the technology begins. The third stage is the control stage where formalized planning and control are introduced. The fourth stage is the integration stage where plans are tailored to include all aspects of the organization. The fifth stage is the data administration stage where the organization has complete shared data and common systems. The sixth and final stage is maturity where data is used as a strategic resource.

Examining database development in terms of Nolan's stages of growth helps explain the proliferation of databases in the DOD. Putting this proliferation in terms of Nolan's stages of growth, the widespread use of databases started in the

contagion stage. In this stage, senior and middle managers became frustrated in their attempts to obtain information from centralized systems. This frustration led to proposals for more local databases. In DOD/DON, the initial emphasis on data centers generated end-user frustration. Data was supplied by the end-users, but access to that data was limited. To resolve this problem many end-users throughout the DOD/DON chain of command started to develop their own specialized databases.

The databases that evolved were influenced by the data model that was in vogue at the time of development. Once these databases were populated, it was perceived as cheaper to maintain, rather than standardizing on one model or format. This was partly due to the view end-users took of their data. Data was viewed as proprietary, not a strategic asset of the entire DOD/DON.

After the explosive growth of databases, the DOD/DON entered the control phase of Nolan's growth model. Here the emphasis is on reduced costs. Redundancies are seen as wasteful. This issue was discussed in appropriations testimony before the House of Representatives on the Corporate Information Management program. One example cited is the DOD payroll systems. Throughout the DOD there are 27 different civilian payroll systems, each with an associated database. These systems range from 25 year old to state-of-the-art technology. As for the Navy, the DON alone had nine systems

in use at the time of the testimony (DOD, 1991, pp.21-22). This example highlights some proliferation problems associated with the contagion stage of Nolan' growth model. As we move further along in the control stage, many of these redundancies will be examined in detail. This is part of the on going Corporate Information Management (CIM) initiative. Often, redesigning and rebuilding systems from scratch to eliminate redundancies is not feasible. Designing systems that can access data already available is a more likely option. However, the requirement is a system that can access and share data across the existing heterogeneous databases. This process has defaulted to a manual one that combines numerous queries across the heterogeneous databases of interest. To avoid the inefficiencies created by this manual process, problems related to homogenizing heterogeneous databases must be resolved. To solve these problems requires an understanding of the different types of heterogeneity that exist in database applications.

B. TYPES OF HETEROGENEITY

Heterogeneity exists at three basic levels. The first is the information systems level. Data is managed by a variety of information systems based on different data models and languages (e.g., file systems, navigational database systems, relational database systems, etc.). The second level of heterogeneity is that of semantics. Since different

information systems have been designed independently, semantic conflicts are likely to be present. This includes both schema (e.g., name, type conflicts) and data (e.g., inconsistencies) conflicts. Finally, the third level of heterogeneity is that of hardware, operating systems, and communications. The three levels of heterogeneity as it applies to database applications are discussed briefly in the following sections.

1. Database Management Systems Heterogeneity

The need to share large amounts of data led to the development of centralized databases and database management systems. The data was grouped by files of records. Each record contained several attributes. Managing the files via a database consisted of three primary tasks, defining the data structure, developing a data manipulation language, and developing a data query language (Parsaye, 1989, pp.40).

The data manipulation and query language depended on how the user perceives the data in the database. The three core models that evolved were the hierarchical, network, and relational models. All three of these data models are still in use.

The hierarchical model is based on the concepts of a tree structure. Each node has branches that point to the children of that node. Every node has a parent except for the root node. Hierarchical databases often exhibit poor flexibility, but have good performance.

The network model is similar to the hierarchical model. However, it uses additional pointers so that links between any nodes can be created. CODASYL is a good example of a network model that developed out of the COBOL language (Gillenson, 1990, pp.256). Both the hierarchical and network models are considered navigational data models which get their power from storage and retrieval techniques.

The relational model uses tables to view data. It is based on the concept that data is organized and stored in two-dimensional tables called relations. Each row in a table represents a record. Each column represents a field. The entire table is roughly equivalent to a file (Kroenke, 1988, pp.132).

These three models represent the foundation of most database management systems (DBMS) in use today. Over time, the need for adding more semantics to the models was recognized. This led to the development of models that tried to capture more semantic information. Chief among these models was the Entity Relationship Model. An entity is a representation of a real world object. Each entity has properties or attributes. Entities in a particular system have symbolically stated relationships.

The latest data model is the object-oriented model. The object-oriented model uses objects to model the domain of interest. The objects have names, attributes, and methods associated with them. Object-oriented databases are gaining

in popularity and the use of the object model as an integrating data model in heterogeneous environments is the focus of this thesis.

2. Data and Schematic Heterogeneity

Since databases are developed independently with different designs, semantic conflicts are likely to occur. Semantic conflicts are classified as either schema or data conflicts.

Schema conflicts occur when different structures or symbology is used to represent the same information, or when a similar structure or symbology is used to represent different information. Schema conflicts include name and structure conflicts. Data conflicts are generally caused by failures to maintain a database or data entry error. These conflicts include violations of databases integrity constraints, the use of different representations for the same data, and inconsistent data. In the next chapter, we present an overview of schematic and data heterogeneity in relational databases.

3. Platform Heterogeneity

> "Heterogeneous computing environments consist of dissimilar hardware or software systems. Because of the diversity, interconnecting systems is far more difficult in heterogeneous environments than in homogeneous environments where each system is based on the same or closely related, hardware and software." (Notkin, 1987, pp.41)

Heterogeneity of hardware is often unavoidable. It occurs in DOD/DON through the acquisition process. As technology evolves, different types of hardware systems are developed that meet the specification of proposals which start the acquisition process. The DOD/DON traditionally goes for the least expensive system that meets the specification without regard to existing architecture (unless existing architecture is taken into account in the specification).

The problems that arise due to hardware and software heterogeneity generally fall under one of the following general areas; interconnection, filing (data storage), authentication, naming, and user interfaces. The following paragraphs give a brief description of each problem.

Interconnection problems deal with how dissimilar systems communicate. Two basic mechanisms for communication are message passing and remote procedure calls. Message passing consists of passing data asynchronously from one process to another. Remote procedure calls provide semantics across a network that are similar to procedure calls in a standard programming language. This type of communication is synchronous in nature. Either of these methods must work with a standard set of communication protocols such as TCP/IP.

The filing problems center on the different data formats used by different computer architectures. An example would be one system using ANSI retrieving a file from a system

using EBCDIC or a system that uses 16 bit words retrieving a file from a system that uses 32 bit words.

The authentication problems deal with the concerns of three broad problem areas: sources of distrust and diversity with respect to authentication; identifying the actual function of authentication and authorization; and accommodating the need for local autonomy within global authentication environments.

The naming problems center on the naming scheme adopted for files or applications. Names come in two types, relative and absolute. An absolute name refers to the same object regardless of its context. This facilitates sharing since a common vocabulary would be implied. A relative name is context dependant. Relative naming has greater utility. Another problem related to naming is the choice of a single global homogenous name space, or many local name spaces. The choice of naming scheme will have a design impact on the development of any multidatabase system.

The final problem area deals with the user interface. Mark Weisner of the University of Maryland defined four levels of user interface heterogeneity; (1) what the user sees, (2) what the application sees and provides, (3) what the window system sees and provides, and (4) what the hardware provides (Notkin, 1987, pp.48-49).

C. PROPOSED SOLUTIONS

There are two general approaches for providing integrated access to a collection of heterogeneous databases. They are the multidatabase or composite approach and the federated approach.

1. The Multidatabase or Composite Approach

The multidatabase or composite approach relies on a global schema. The global schema provides a description of the information in the heterogeneous composite databases and make up a logically single, integrated database. Access and manipulation operations are expressed in a universal query language and mediated through the global schema. This format provides the user with the illusion of a centralized database. (Collet, 1991, pp.50)

Construction of a global schema is a difficult process. The main reason is the lack of a general solution for the semantic conflicts in a situation in which the autonomy of each of the constituent databases is preserved (Litwin, 1986, pp.213). Furthermore, the process must be repeated every time a composite database schema changes or another composite database is added to the system.

The users are not required to know what semantic conflicts exist among the composite databases. However the developers must provide explicit resolutions for the conflicts before actual system use. In essence a centralized view of

all the composite databases is developed. This centralized or virtual view may be different from the local views of the composite databases. The view discrepancy can cause problems in the execution of existing applications.

2. The Federated Approach

By contrast with composite or multidatabase systems, the federated database uses an organization model based on equal, autonomous databases, with sharing controlled by explicit interfaces (Heimbigner, 1985, pp.48). The user is shown a collection of local views along with tools for information sharing among the composite databases. In essence, a virtual global schema is created.

Federated databases try to minimize central authority, yet support partial sharing and coordination among composite databases. Without the constraint of a central authority the federated system tries to maintain as much composite database autonomy as possible and still support strong information sharing.

To facilitate the conflicting requirements of autonomy and data sharing, the federated architecture relies on three component schemas: private schema, export schema, and import schema. The private schema is the schema that describes a composite database and is stored at the location of the composite database. The export schema is the portion of the schema that a particular composite database is willing to

share. The import schema specifies the information that composite databases desire to use from other composite databases. (Heimbigner, 1985, pp.54)

Negotiation is another key feature of the federated architecture. This system is conceptually made up of two parts; an interpreter, and a collection of procedures written in the negotiator's language. This negotiation aspect is where most of the heterogeneous conflicts are resolved.

D. INTEGRATING MODEL REQUIREMENTS

Either approach requires a strong integrating model that is semantically rich enough to subsume the component databases. The composite or multidatabase needs a semantically rich model to build an all encompassing global schema. The federated model needs a semantically rich model to supply the needs of its negotiator. This thesis uses the object oriented model as the integrating model and develops a framework for representing the semantic heterogeneity for this model.

III. SCHEMA AND DATA CONFLICTS IN THE RELATIONAL MODEL

This chapter is a synopsis of an article by Won Kim and Jungyun Seo from the December 1991 issue of Computer magazine called "Classifying Schematic and Data Heterogeneity in Multidatabase Systems." The article developed a complete framework for enumerating and classifying the types of multidatabase system structural and representational discrepancies.

When viewed in a relational sense, the schema conflicts can be categorized in three main area's: table-versus-table conflicts, attribute-versus-attribute conflicts, and table-versus-attribute conflicts. Each of these can be further broken down. We will look at each schema conflict subarea separately, and then discuss the data conflicts.

A. TABLE-VERSUS-TABLE CONFLICTS

The table-versus-table conflicts occur when different databases use different definitions to represent similar information in tables. Table versus table conflicts can be categorized as one-to-one and many-to-many table conflicts. One-to-one table conflicts occur when the different databases represent similar information using different names, structures, and constraints in single tables. The table name

conflicts arise when different names are used in different databases to represent semantically equivalent tables. An example would be one table named "document" that describes real world paper-media documents, and another database with a table "publication" that describes the same object. A second version of this conflict occurs when different databases use the same table name to represent semantically different tables. Going back to the document example, we compare this to another database that has a table named "document", yet the attributes describe those of a file on a harddrive in a computer versus a paper media. The table structure conflicts occur when different databases have similar tables, yet the numbers of attributes in the tables differ. The table constraint problem arises from differences in the specifications of the tables in the different databases. These problems are associated with the use of primary, candidate, and foreign keys. If an attribute is a key in one database, but the corresponding attribute in the table of another database is not a key, it is difficult to impose constraints on this attribute at a homogenizing level. Thus, making updates based on a key with a conflict is difficult.

Many-to-many table conflicts occur when different databases use a different number of tables to represent the same information. This type of conflict can usually be decomposed into one-to-one table conflicts.

B. ATTRIBUTE-VERSUS-ATTRIBUTE CONFLICTS

The attribute-versus-attribute conflicts occur when different definitions for semantically equivalent attributes exist in different databases. Like table conflicts, one-to-one and many-to-many attribute conflicts exist. Many-to-many conflict, however, can be decomposed into one-to-one conflicts. The attribute-versus-attribute conflicts can be categorized as attribute name conflicts, default value conflicts, and attribute constraint conflicts.

Attribute name conflicts arise from using different names for semantically equivalent attributes in different databases or when the same attribute name is used for semantically different attributes. This is very similar to the table name conflicts described earlier.

Attribute default value conflicts arise when one database enters a null when no attribute value is entered, while another database enters another default value when no value for the same attribute is entered.

Attribute constraint conflicts fall under two types; data type conflicts and attribute integrity-constraint conflicts. Data type conflicts occur when semantically equivalent attributes in different databases have different data types. An example would be an attribute representing a social security number stored as a numeric type in one database, and as a character type in another database. Attribute integrity-constraints conflicts are similar to default value conflicts.

Specifically, they deal with the field size of an attribute. An example would be the attribute weight in one database being defined as an integer less than 999, while the same attribute is defined as an integer less than 9999 in another database. This would cause a problem in homogenizing the two databases when a four digit value is entered. It would work for one database, but not the other.

C. TABLE-VERSUS-ATTRIBUTE CONFLICTS

The third category is table-versus-attribute conflicts. These conflicts occur if one database uses tables while another uses attributes to represent the same information. Often this conflict type can be regarded as a combination of many-to-many table conflicts and many-to-many attribute conflicts.

D. DATA CONFLICTS

In relational models the data conflicts fall into two subcategories: data conflict that violate specified integrity constraints and conflicts based on different representations for the same data. The first problem can be expressed as wrong data. This is generally caused by a failure to maintain a database or a failure to enforce integrity constraints. We see this problem when equivalent attributes of different databases are expected to have the same value, yet the values are different. Another common cause of this problem is

obsolete data. This can be seen when two databases have similar data, yet one has more frequent update periods. An example would be two similar databases that track individual flight hours. One is updated weekly and the other is updated monthly. If each database was queried for LT Smith's flight hours, the results are likely to be different.

The second type of data conflict, different representations for the same data can actually be viewed in three different aspects. The first deals with different expressions. This occurs when the same type of data has different expressions in different databases. An example would be listing LT Smith's rank as "LT" in one database and "0-3" in another. In USN terms both mean the same thing. The second aspect deals with different units. These conflicts arise when different databases use different units for similar numeric data. An example of this would be a flight time database that uses minutes as the measurement of flight time, while another flight hour database that uses hours and partial hours to record the flight time. The third aspect is different precision. Precision conflicts occur when two similar databases use values from domains of different cardinalities for the same attribute. An example would be one data base that uses light, medium, and heavy to describe the weight of an aircraft, while another uses a numeric range of 100 to 200,000 pounds to describe weight. Figure 1 is a

synopsis of the schema and data conflict classifications.(Kim, 1991, pp.12-18)

E. CONCLUSION

The objective of this thesis is to develop a similar framework for classifying schematic and data conflicts in an object-oriented model. The next chapter introduces the object-model used in support of this endeavor.

I. Schema Conflict

A. Table-versus-table conflicts

1. One-to-one table conflicts

a. Table name conficts

1) Different name for equivalent tables

2) Same name for different tables

b. Table structure conflicts

1) Missing attributes

2) Missing but implicit attribtes

c. Table constraint conflicts

2. Many-to-many table conflicts

B. Attribute-versus-attribute conflicts

1. One-to-one attribute conflicts

a. Attribute name conflicts

1) Different names for equivalent attributes

2) Same name for different attributes

b. Default value conflicts

c. Attribute constraint conflicts

1) Data type conflicts

2) Attribute integrity-contraint conflicts

2. Many-to-many attribute conflicts

C. Table-versus-attribute conflicts

II. Data Conflicts

A. Wrong data

1. Incorrect-entry data

2. Obsolete data

B. Different representation for the same data (Same representation for different data)

1. Different expressions

2. Different units

3. Different precisions

Figure 1 Schema and Data Heterogeneity Conflicts in Relational Models

IV. THE OBJECT-ORIENTED MODEL

The use of an object-oriented model gives us richer semantics and greater modeling power over alternate approaches. Additionally, an object-oriented model is an ideal integration model for combining heterogeneous databases.

A. MANAGING COMPLEXITY

An object-oriented model is used for representing and managing complexity in a problem domain. Although there is no general consensus on what constitutes an object-oriented model, there are some agreed-upon characteristics that give an object-oriented model its semantic richness(Brown, 1991, pp.20). These characteristics include data and procedural abstractions, encapsulation, inheritance, associations, communication via method connections, and function overloading.

1. Abstraction

There are two types of abstraction, procedural and data. Procedural abstraction is the principle that any operation that achieves a well defined effect can be treated by its users as a simple entity, despite the fact that the operation may actually be achieved by some sequence of lower-level operations. Data abstraction is the principle of

defining a data type in terms of the operations that apply to the object with the constraint that the values of such objects can be modified and observed only by the use of the operations (Coad, 1991, pp.14).

2. Encapsulation

Encapsulation is a facility that serves to protect some part of a program or data against improper access. Central to the object-oriented model is the concept that the entities of interest in the real world can be modeled most effectively by representing each real-world entity as an object in the model. The definition of such an object includes both the data properties of that object and the operators which are permitted to manipulate that object. The essence of encapsulation is that such operators form an interface to objects which provide the only way to amend the state of the objects. The user of an object has no way to access that object other than through the defined set of operators (Brown, 1991, pp.19). Encapsulation is often used to enforce information hiding. The power of encapsulation is that it keeps related content together.

3. Inheritance

Inheritance is a mechanism for expressing similarity among classes, and simplifying definitions of classes similar to those previously defined. In general, we find that a subclass hierarchy can be defined in which a subclass is a

specialization of its superclass in the hierarchy. An important aspect of this specialization is that we do not need to define each subclass from scratch. We think of a subclass as inheriting the behavior of its superclass (Brown, 1991, pp.20). This inheritance portrays generalization and specialization making common attributes and functions explicit within class hierarchy. Inheritance allows for the explicit expression of commonality. (Coad, 1991, pp.15)

4. Association

Association is the ability to tie together certain things that happen at some point in time or under similar circumstances. In constructing any type of information model, we are concerned with identifying associations between things in the real world and reflecting those associations as precisely stated relationships in the model (Shlaer, 1988, pp.47). To have an association is to have some logical connection.

5. Communication and Method Overloading

Communication with messages is a principle for managing complexity, especially for interfacing different objects. This communication takes the form of producing functions (or methods). Message data is passed to an object, the data in the message causes reactions with the object. These reactions can be thought of as methods. Methods with the same name can cause different reactions depending on the

amount and type of data supplied in the message. This allows for method overloading.

B. THE BUILDING BLOCKS

1. Class and Objects&Class

An object is an abstraction of something in a problem domain, reflecting the capabilities of a system to keep information about it, interact with it, or both (Coad, 1991, pp.53). Another way of looking at an object, is that it is an encapsulation envelope. It encapsulates knowledge in the form of attribute values and exclusive methods that can be performed with or on the object. [1]

Class is a description of one or more objects with a uniform set of attributes and methods, including a description of how to create new objects in the class. A class of objects contain common traits or attributes and have the same behavior. Figure 2 shows the symbology used to represent an object&class and a class.

The top part in either object&class or the class symbol contains the name of the object or class. This name is a noun that describes the basic concept of the object. The middle area of the object&class or class symbol contains the attributes of the object&class or class. The bottom area of

---

[1] The object model used in this paper is based on the model proposed by Coad and Yourdon in "Object-Oriented Analysis", Yourdon Press, 1991

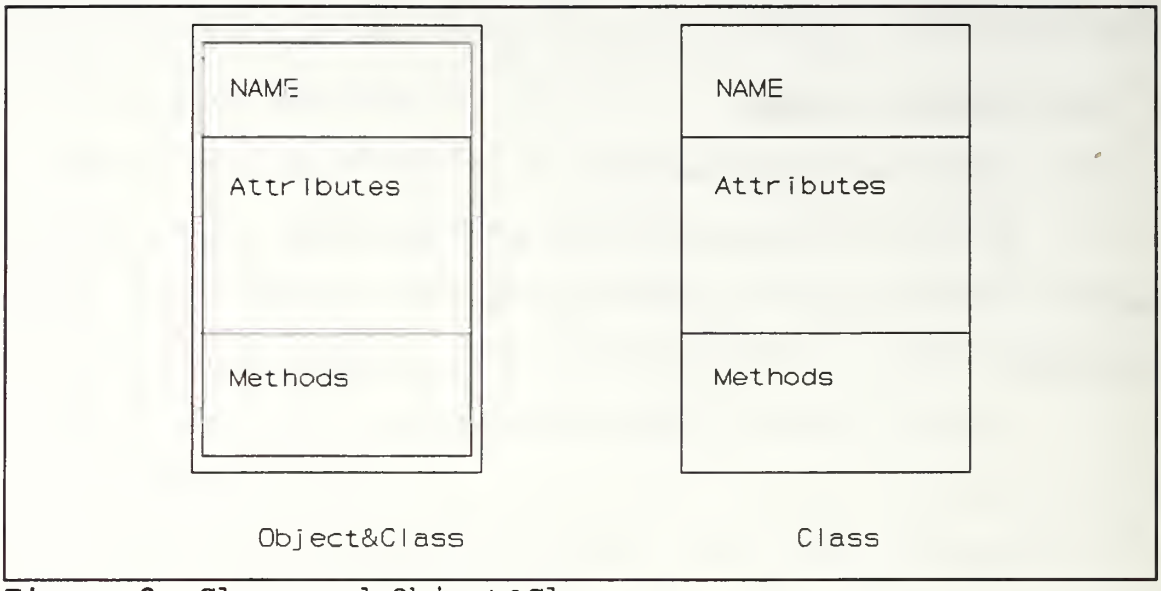

Figure 2 Class and Object&Class

the symbol contains a list of the methods associated with the object&class or class.

2. Structure

The structure is a symbolic expression of the problem domain complexity, pertinent to the systems responsibility (Coad, 1991, pp.78). It indicates the relationships among the object&classes and classes. In this model two types of structure exist. Generalization-specialization structure and whole-part structure.

Generalization-specialization structure is used to distinguish between similar but not identical classes. The attributes and methods germane to the actual class are inherited in the specialization class. The generalization-specialization structure allows for a method of organization

that implies inheritance from generalization class to specialization class and allows for an explicit representation of more attributes and methods pertinent to the specialization class. This structure notation is shown in Figure 3.

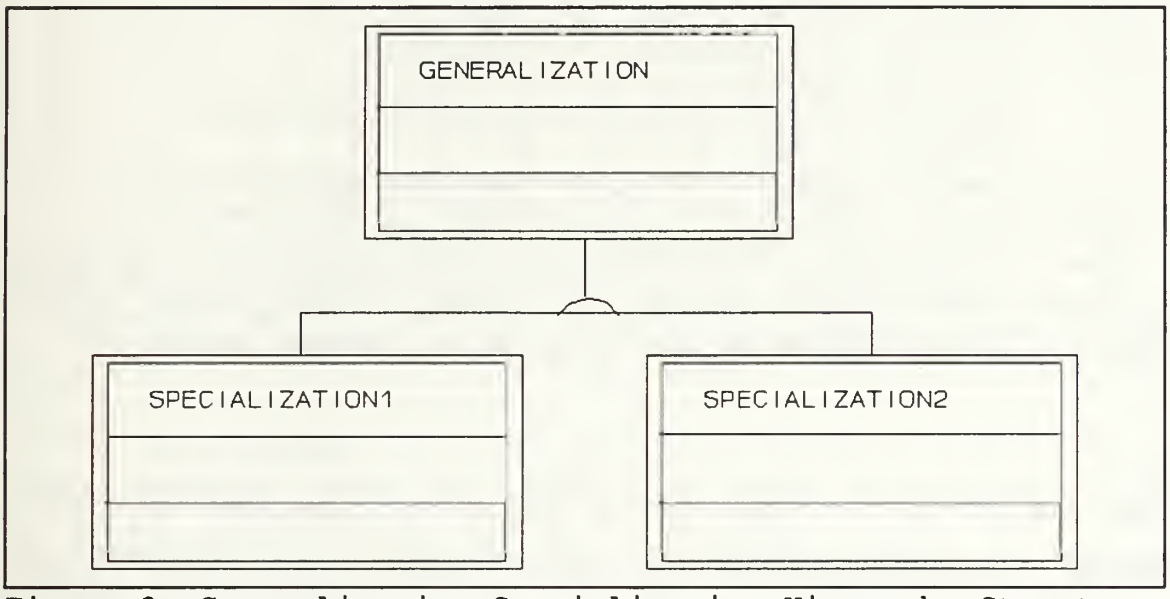

Figure 3 Generalization-Specialization Hierarchy Structure

Their are two types of generalization-specialization structure forms. The structure forms are either a hierarchy (as depicted in Figure 3) or a lattice. Though the hierarchy form is the most common, the lattice structure can capture more information. Specifically, the lattice structure can highlight additional specializations and explicitly capture commonality while only modestly increasing model complexity (Coad, 1991, pp.89). This notation is depicted in Figure 4.

The whole-part structure is based on a basic method of organization. It groups a whole object with the parts of that object. An example of this would be a whole object called

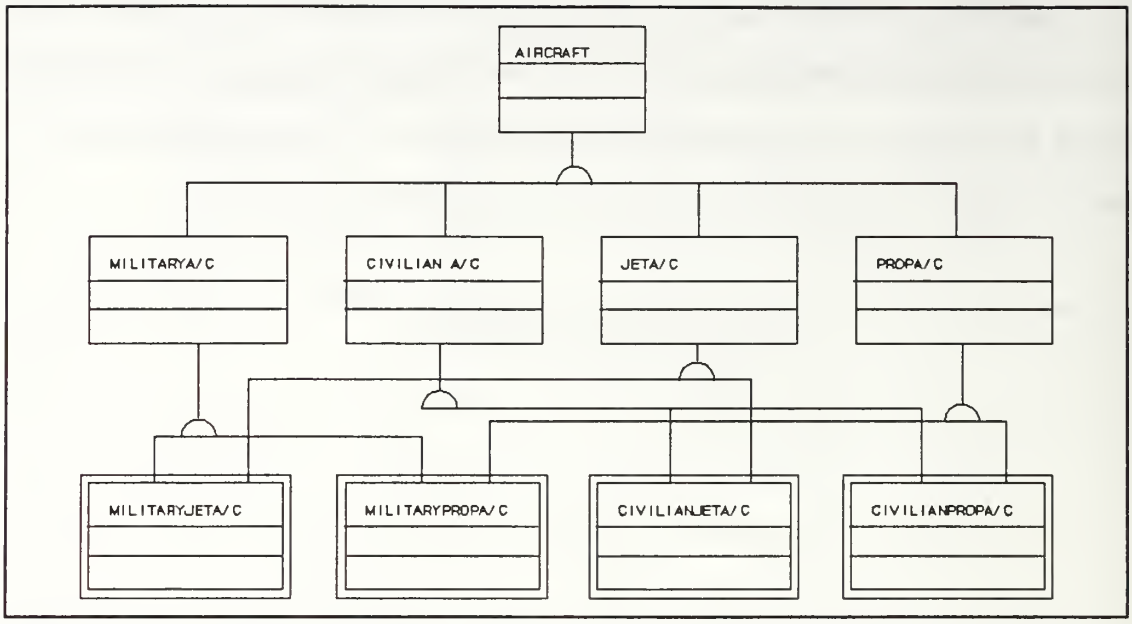

Figure 4 Generalization-Specialization Lattice Structure

ship, associated with a number of part objects like propulsion-plant and cargo. The notation to represent a whole-part structure displays directionality and explicitly the number of parts related to the whole. The notation is depicted in Figure 5.

The term multiple structures is used to describe combinations of general-specialization and whole-part structures. The essence of structure is that structure is an expression of problem-domain complexity pertinent to the system's responsibility. Structure is used as an overall term describing both generalization-specialization and whole-part structures (Coad, 1991, pp.99).

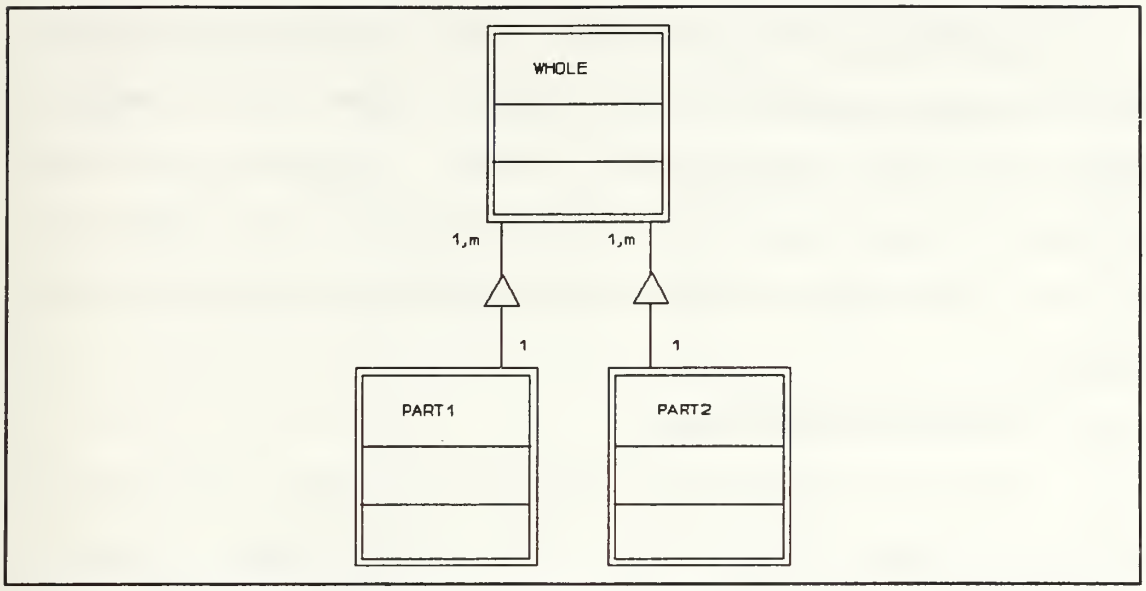

Figure 5 Whole-Part Structure

3. Attributes

Attributes add detail to the class, object&class and structure abstractions. An attribute is some data or store information for which each object in a class has its own value. Attributes may only be changed by exclusive methods. If another part of the system needs to manipulate an attribute of an object, it must specify a message connection that contains information to a method defined by the object (Coad, 1991, pp.120).

Each attribute of an object must capture a complete compact concept. This concept must be important to the problem domain. Making each attribute a complete compact concept reduces the number of attributes that must be included in the object. This leads to a simpler model for review.

Attributes of a generalized class in a generalization-specialization structure also apply to the specialization objects of that generalized class. Determining were attributes should be placed in a structure is an important part of determining the generalization-specialization structure.

4. Instance Connections

Instance connections model associations. An instance connection models the problem domain mapping that one object needs with other objects in order to fulfill its responsibilities. These can be one-to-one instance connections causing a mandatory association between one object and another, or optional association, or mandatory one way but optional in the other direction. One-to-many, or range of possibilities, is also captured in instance connection symbology.(Coad, 1991, pp.126) An instance connection is modeled in Figure 6.

5. Methods

A method is a specific process that uses data from an object. Up to now we have discussed how to model data. Methods are how we model processes.

Each object exists in different states. The state of an object is reflected by the values of it's attributes. Methods are the processes that change the values of the attributes. This implies that knowledge about the state of an

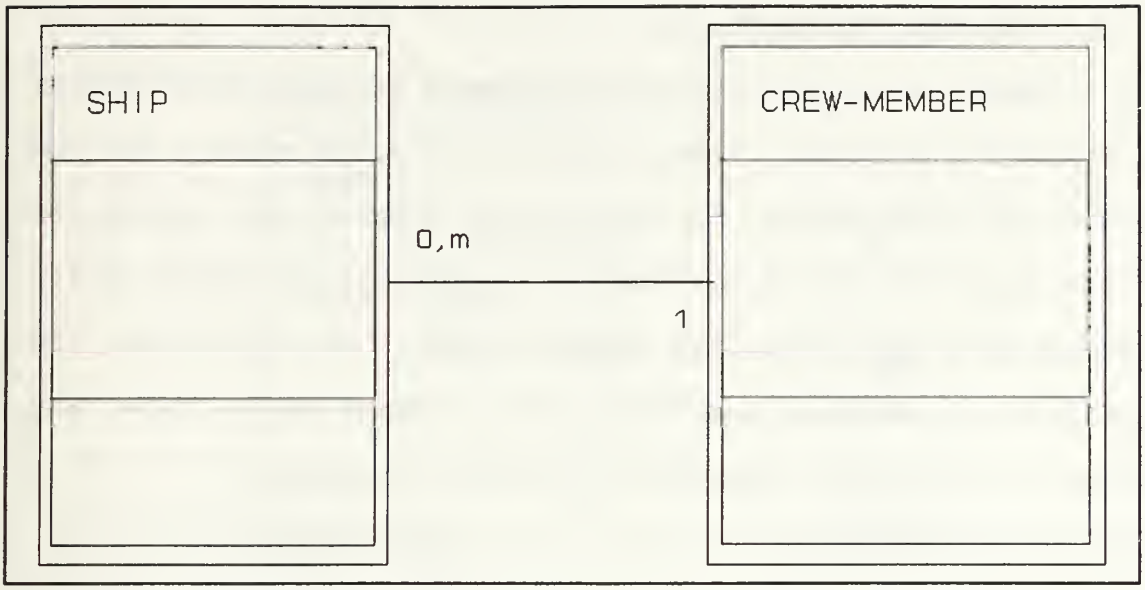

Figure 6 Instance Connection

object is important when examining methods.

The methods of a system can be divided into two main types; algorithmically-simple methods and algorithmically-complex methods. Algorithmically-simple methods apply to each class and object in the model. They are create, connect, access, and release methods. The create method is used to create and initialize a new object in a class. The connect method connects or disconnects an object with another object. The access method sets the attribute values of an object. The release method deletes an object. The algorithmically-complex methods fall into two categories. The calculate category methods use attribute values to calculate specific results. The monitor category methods monitor external systems or devises.

6. Message Connections

Message connections are the means of connecting object to facilitate methods. These connections exist solely for the benefit of the methods. Each message connection represents values sent within the context of a particular method and a response as a result of that method (Coad, 1991, pp.155). The notation for a message connection is a dashed arrow connecting objects or a class to objects as shown in Figure 7.

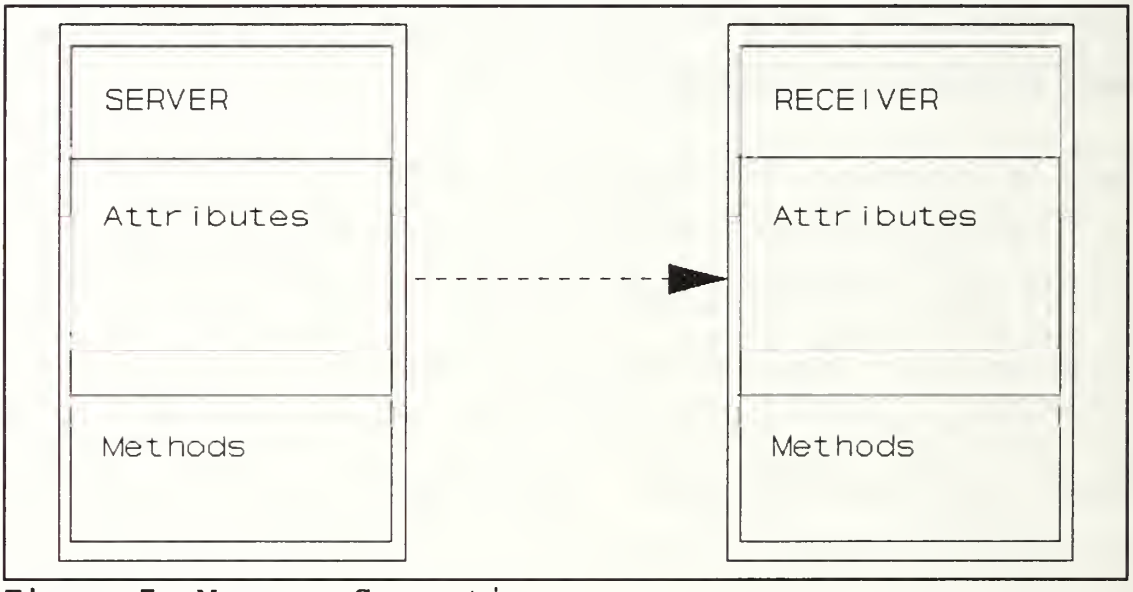

Figure 7 Message Connection

Additionally, one message connection sender object can send a message to multiple receiver objects. The values sent in the message connection invoke methods in each object that receive the connection. The annotation is shown in Figure 8.

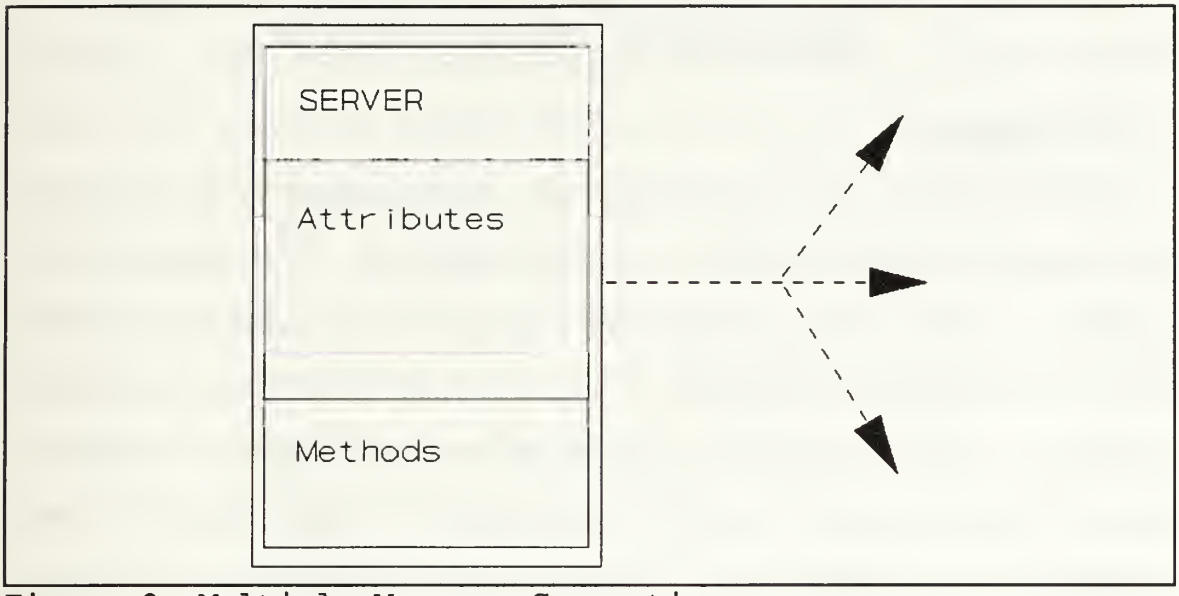

Figure 8 Multiple Message Connections

C. CONCLUSION

This chapter explained the tools used to analyze the problem domain in the framework of object-oriented analysis. The notations used were adopted from "Object-Oriented Analysis" by Peter Coad and Edward Yourdon. These tools give us a strong modeling power, and support the semantics needed to represent the semantics of many data models. These concepts and tools are used in the next chapter to transform schemas of databases based on several data models into equivalent schemas in an object-oriented model in preparation for integration into a global schema.

V. HETEROGENEOUS DATABASE SCENARIO

A. BACKGROUND

The database scenario used in this thesis comes from a U.S. Navy Maritime Patrol Training Squadron. The squadron is the east coast training squadron for all P-3C maritime patrol aircraft aircrew positions. We examine four heterogeneous databases that have been independently developed to support various applications of the squadron. Specifically, the databases include a library database for classified material, an aircrew physiology database, a Naval Air Training, Operations and Standardization (NATOPS) department database, and a flight schedule database.

These databases were developed by different people and at different times using different data models. They have never been standardized in terms of data elements. All of them run on personal computers. They were developed with either "Q and A", "Dbase III plus", or "Enable". The command has interest in developing more databases, but lacks the expertise to design and implement an overall database application that covers all of their needs.

Each of the databases being examined has a specific purpose. The library for classified material database maintains data on all classified material used for

instruction. The flight physiology database tracks the status of student and staff aircrewmen in regards to physiology, survival, and other expiring qualifications. The NATOPS department database tracks NATOPS qualifications and NATOPS publications issued to all aircrewmen. The flight schedule database is used to promulgate a daily flight and ground training events' schedule. Frequently, queries that span across these databases need to be answered. Currently, this is accomplished manually through a tedious procedure. First, the database that contain the data to be accessed are identified. Second, several queries in different languages are formulated and executed on the different databases. Third, the results are transferred to the requesting site, combined, and the requested information extracted and formatted. Additionally, considerable overlap occurs among the four databases.

To allow queries that span several databases, a federated approach is suggested. With this approach, each local database is considered a logical component in the federation (Heimberger, 1985, pp.48). The components are tied together by explicit interfaces that form a virtual global schema that represent the integration of the local schemas. To accomplish this several steps are necessary. First, each local schema is transformed into an equivalent schema in a semantically rich common data model. Second, schema conflicts need to be identified and resolved. Third, the local schemas in the

common data model are merged to form a virtual global schema. Fourth, an additional control component, known as the global controller, is required. The global controller maintains the definition of the virtual global schema and acts as a coordinator and translator: it receives a global query, possibly in a user specific language; translates it into an equivalent query on a common-model global schema; decomposes and translates the common-model query into subqueries to the corresponding local database sites for processing; collects the results; identifies and resolves data content conflicts; reformats the results; and sends it back to the originating site.

B. THE LIBRARY FOR CLASSIFIED MATERIAL DATABASE

The library database contains the data necessary to track classified document that are issued to students and instructors while under a course of instruction at the training squadron. The data is grouped by different components: the library data includes the name of the library and the custodian; document data includes the publication name, document number, classification, status (checked in or out), and if status is checked out the social security number of the document holder; student data includes name, social security number, locker number, secret folder numbers, class number, and crew number; staff data includes name, social

security number, locker number, secret folder numbers, safe number, and crew number.

Each document has a serial number; each student has an assigned locker and secret folder number; and each instructor has a secret folder number. Documents classified below the secret level are issued to students. The students store them in their confidential lockers when not in use. Secret documents are issued to students, but stored in student secret folder in the IML vault. The students check out their folders when they need documents for class, study, or flights. Instructors can check out confidential documents and store them in approved safes. Additionally, they can check out secret documents and store them in approved safe or use a secret folder in IML. The choice for instructors comes down to a matter of convenience; however, all applicable security precautions apply.

The IML staff conducts a daily inventory of all secret material. This is conducted at the end of the normal work day. This inventory includes all secret material not issued, all student secret material, and all instructor material stored in IML's vault. Instructor material stored in individual safes are periodically inventoried. All other student material is inventoried upon check-in and check-out. This occurs every six weeks.

Problems that arise are usually related to the flight schedule. Often, an individual has material signed out, and

is on a flight or trainer that is scheduled to land or finish after normal working hours. This material is stored in a separate safe and inventoried the next morning.

1. Classified Library Relationship Diagram

The classified library relationship diagram is depicted in Figure 9.

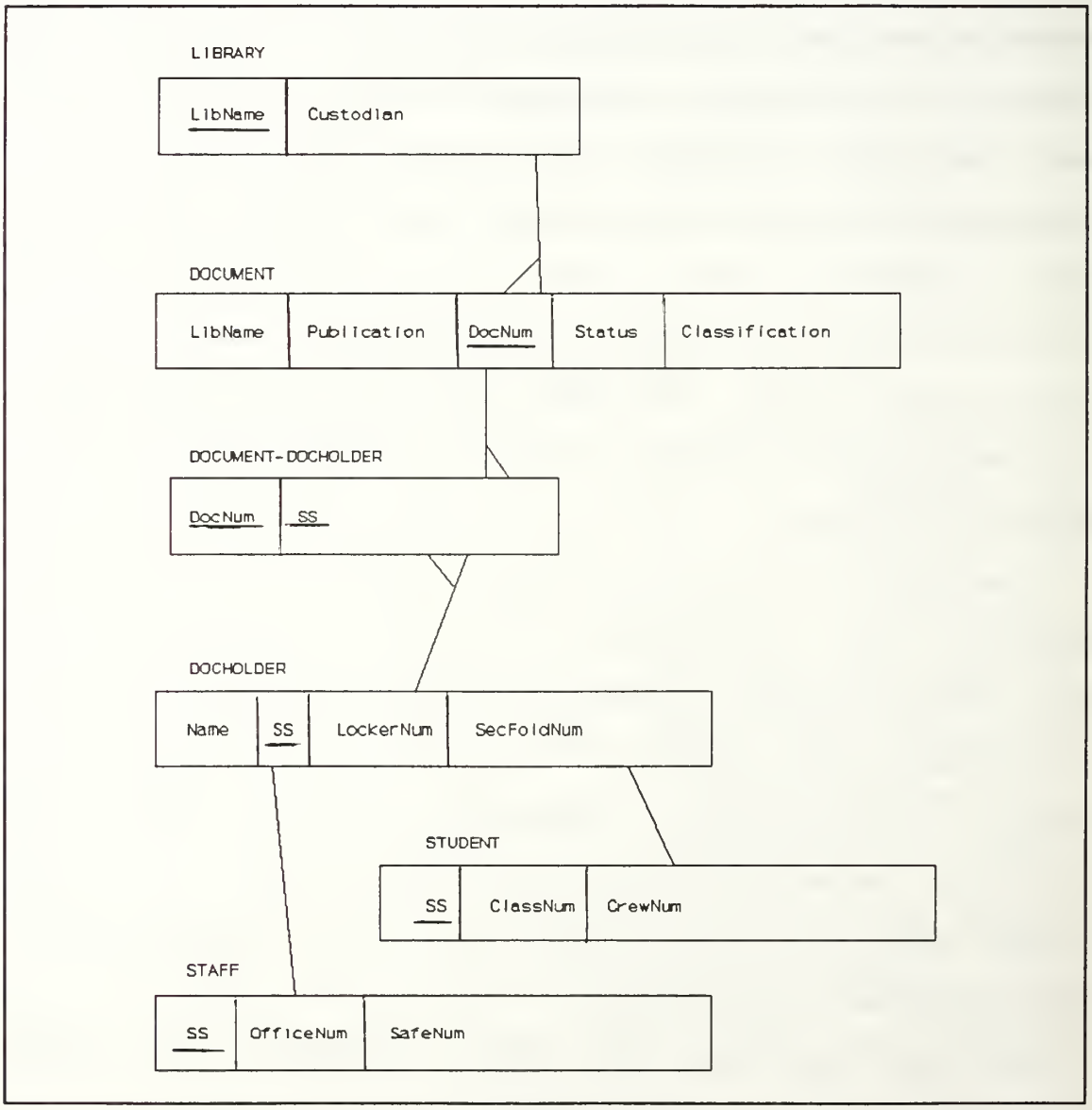

Figure 9 Library of Classified Material Relational Diagram

2. Classified Library Data Dictionary

NAME	TYPE(Length)	Description
LibName	Character (30)	Name of Library
Custodian	Character (35)	Name of Custodian
Publication	Character (50)	Title of Document
DocNum	Numerical (5)	Assigned Serial Number
Status	Character (4)	In or out of library
Classification	Character (6)	Conf, Secret, or None
Name	Character (25)	Name and Rank of Person
SS	Numeric (9)	Social Security Number
LockerNum	Numeric (3)	Assigned Locker Number
SecFolderNum	Numeric (3)	Assigned Secret Folder
ClassNum	Numeric (4)	Assigned Class Number
CrewNum	Character (2)	Assigned Crew Number
OfficeNum	Character (3)	Office Number
SafeNum	Character (3)	Safe Number

3. Transformation Process

The transformation process is started by examining each table to see if it could be modeled as an object. Most objects are either tangible things, roles, incidents, interactions, or specifications (Shlaer, 1988, pp.14).

The easiest objects to identify are the tangible things. Library, document, and document holder fit in this category. Student and staff also fit, but they are specific types of document holders. They are modeled as

specialization objects to the generalization object document holder. The Document-DocHolder relationship is not an object but conveys a necessary relationship that must be modeled.

After determining objects, we list attributes associated with each one. Next, we examine the application to determine the methods associated within each object. The implicit methods of add, edit, and delete are not modeled. They are implied in the object class structure. The methods that must be modeled are library inventory, custody reports, individual inventory, check-in and check-out.

4. The Classified Library Object Model

The classified library object model is depicted in Figure 10.

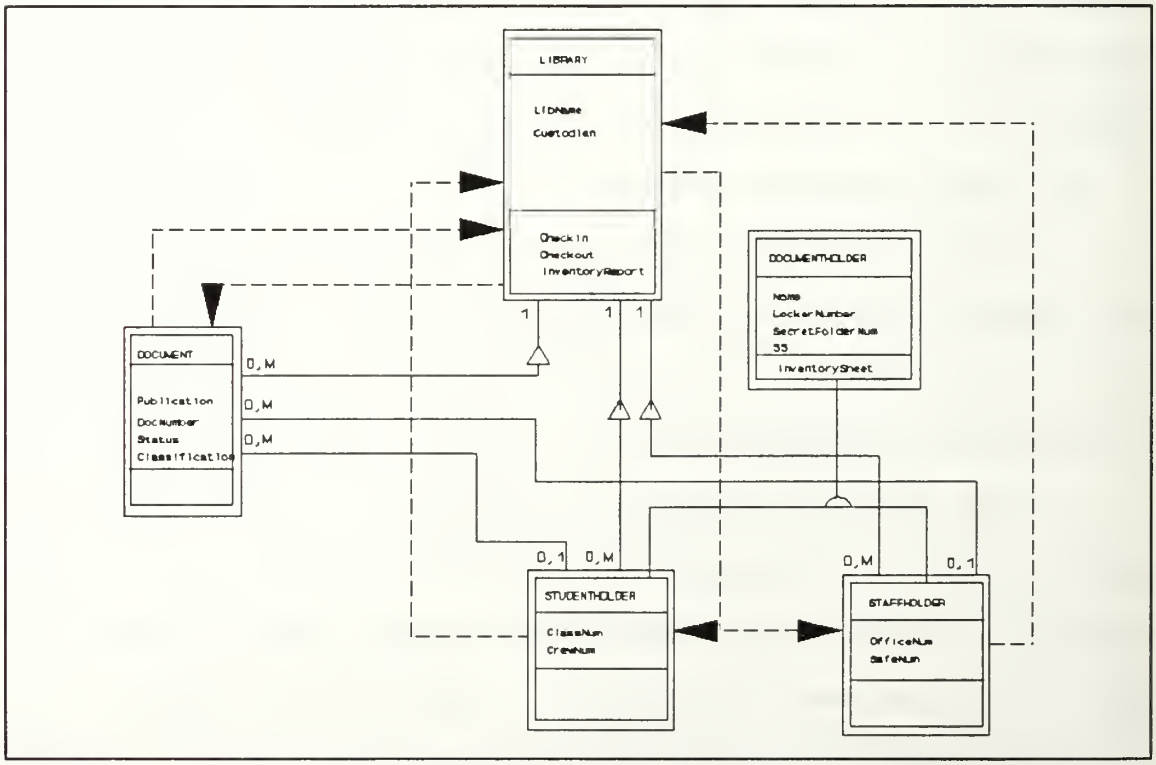

Figure 10 Classified Library Object Model

C. THE FLIGHT PHYSIOLOGY DATABASE

The flight physiology database consists of data on officers, enlisted personnel, and aircrew qualification requirements. The data is grouped by the different components: officer data includes name, rank, social security number, birth month, and designator (pilot or flight officer); enlisted data includes name, rate, birth month, and social security number; requirement data includes type of requirement and the date it is due.

Every naval aircrew member must have certain expiring qualification to continue flying. These include flight physicals, basic survival swim qualifications, advanced water survival (DWEST), flight physiology training, instrument qualifications, SERE, and NATOPS qualifications. The aircrew position determines which events are required and how often. If a required qualification lapses, the particular aircrew member is considered in a down status until that qualification is obtained. While in a down status he cannot perform his normal aircrew duties.

1. Flight Physiology Relational Diagram

The flight physiology relational diagram is depicted in Figure 11.

2. Flight Physiology Data Dictionary

NAME	TYPE(Length)	Description
LastName	Character (25)	Last Name of Individual

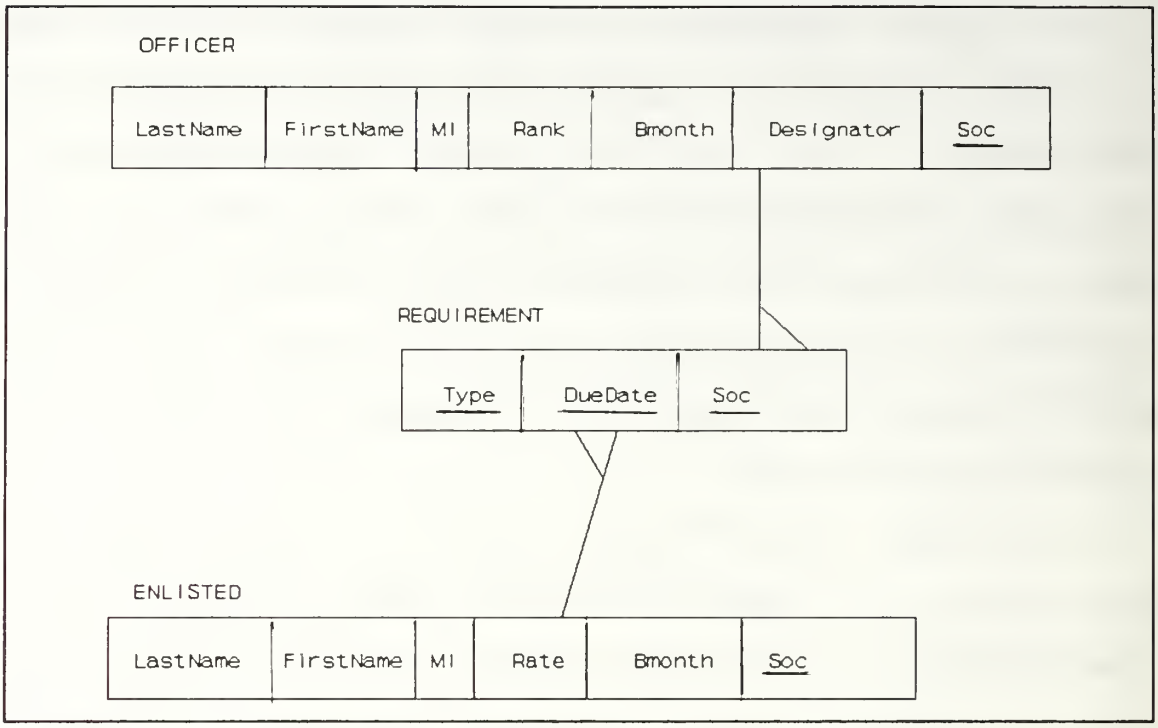

Figure 11 Flight Physiology Relational Diagram

FirstName	Character (25)	First Name of Individual
MI	Character (1)	Middle Initial
Rank	Character (5)	Rank
Bmonth	Character (3)	Birth Month
Designator	Character (4)	Designator of Officer
Soc	Character (9)	Social Security Number
Name	Character (15)	Type of Qualification
DueDate	Date	Date Qual Complete
Rate	Character (7)	Rate of Enlisted
Crewnum	Character (4)	Number of assigned crew

3. Transformation Process

All of the tables can be transformed to objects, however, officer and enlisted share a number of attributes. This similarity is captured by using a generalization-specialization structure. Creating a class to capture the similar attributes is used. This class is labeled "servicemember" and has the attributes lastname, firstname, mi, and soc. The specialization objects of this class are "enlisted" and "officer".

The methods are derived from the applications of the database. The database is used for planning inputs to a master schedule and to notify individuals of expiring qualifications. These methods are listed as "planninglist" and "notification".

4. The Flight Physiology Object Model

The flight physiology object model is depicted in Figure 12.

D. THE NATOPS DEPARTMENT DATABASE

The NATOPS department database consists of data on officers, enlisted personnel, and NATOPS qualification requirements. The data is group by the different components: aircrew data includes name, rank, social security number, position, and crew number; test and check flight data includes type of test or check flight, date of item, test or flight administer, and score; publication data includes name, number,

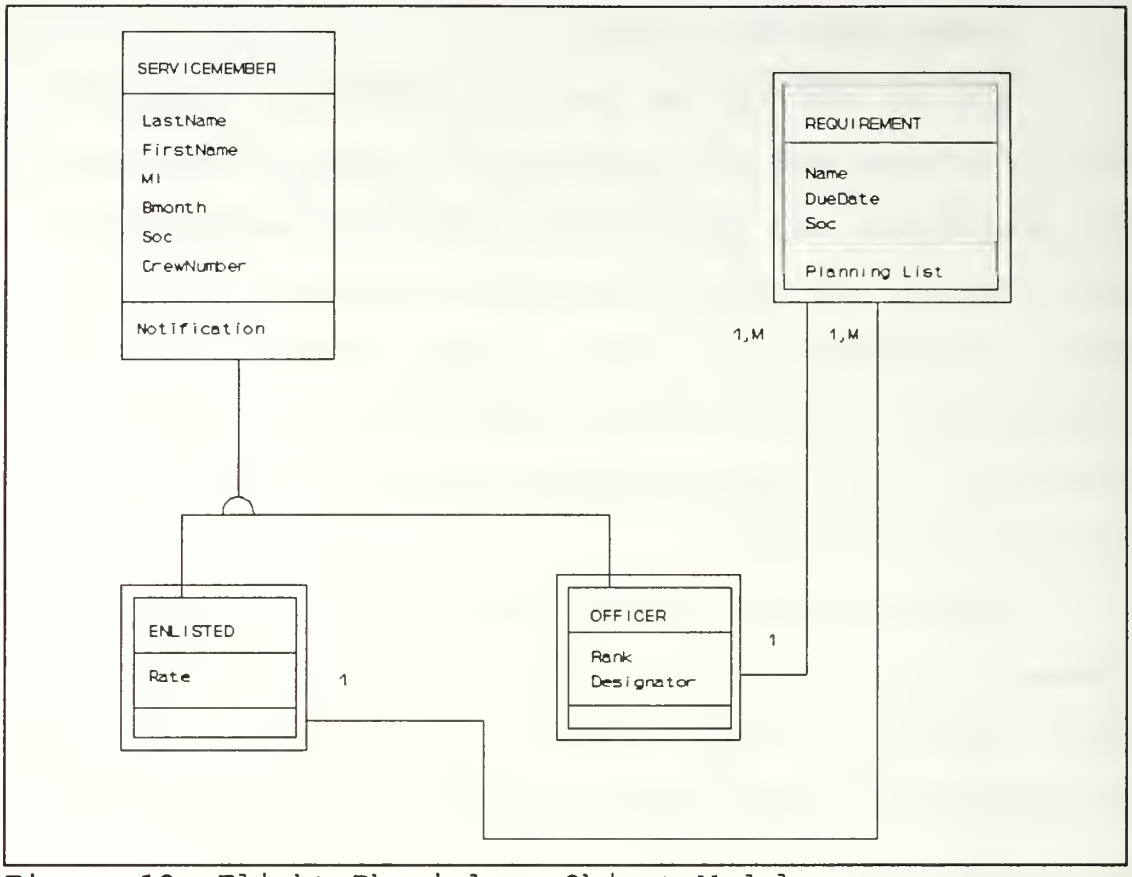

Figure 12 Flight Physiology Object Model

and current change number.

This database is used to track the NATOPS qualification progress of aircrew. It keeps track of open and closed book test scores, oral exam scores, and flight grades. Each aircrew student is associated with an aircrew position. That position is associated with required tests, oral exam and flights. Additionally, the department tracks NATOPS publications issued to all squadron aircrew.

Two primary applications are associated with this database. In the first application, publications are tracked for accountability. Additionally the database assists in

recall purposes when updates to publications are required. The second application is generating a 90 day planning tool where all personnel needing renewed or initial NATOPS qualifications are tracked 90 days before their due date. This tool is used as an input to a monthly planning calendar.

1. NATOPS Department Relational diagram

The NATOPS department relational diagram is depicted in Figure 13.

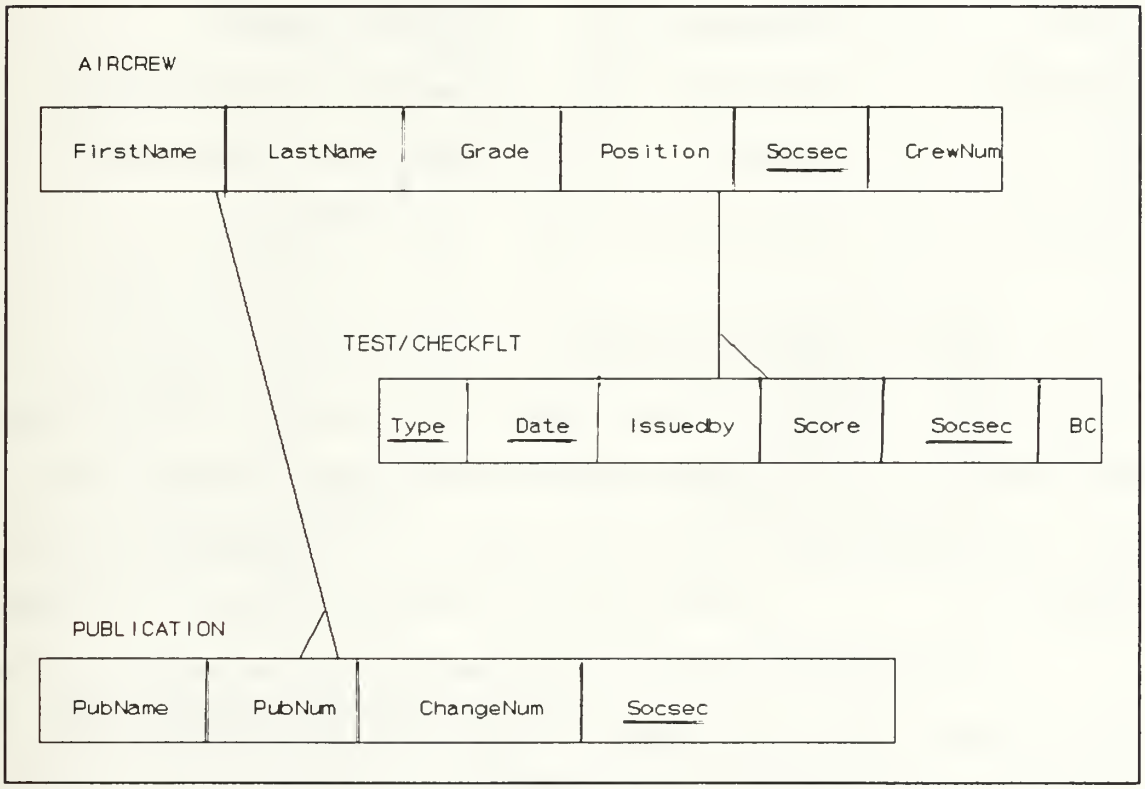

Figure 13 NATOPS Department Relational Diagram

2. NATOPS Department Data Dictionary

NAME	TYPE(Length)	Description
LastName	Character (25)	Last Name of Individual

FirstName	Character (25)	First Name of Individual
Grade	Character (3)	Paygrade
Position	Character (3)	Crew position
Socsec	Character (9)	Social Security Number
Type	Character (6)	Open, Closed etc.,
Date	Date	Date Obtained
Issuedby	Character (25)	Name
Score	Numeric (4)	Numeric score obtained
PubName	Character (35)	Name of Publication
PubNum	Character (6)	Serial Number
ChangeNum	Character (4)	Latest change entered
BC	Logical	Blue Card Holder Y/N
CrewNum	Character(4)	Crew Number

3. Transformation Process

All of the tables can be modeled as objects. Additionally, none of the tables contain similarities. So, to make the transformation each table is transformed into an object.

In determining the methods we examine the applications that access this database. One primary application is maintaining a publication inventory list. The second application is generating a 90 day planning input for NATOPS qualifications that expire in the next 90 days. Additionally a third application is sending out change notices for

publications, and tracking change entries into applicable NATOPS related publications. Finally the last application is tracking individual performance.

4. NATOPS Department Object Model

The NATOPS department object model is depicted in Figure 14.

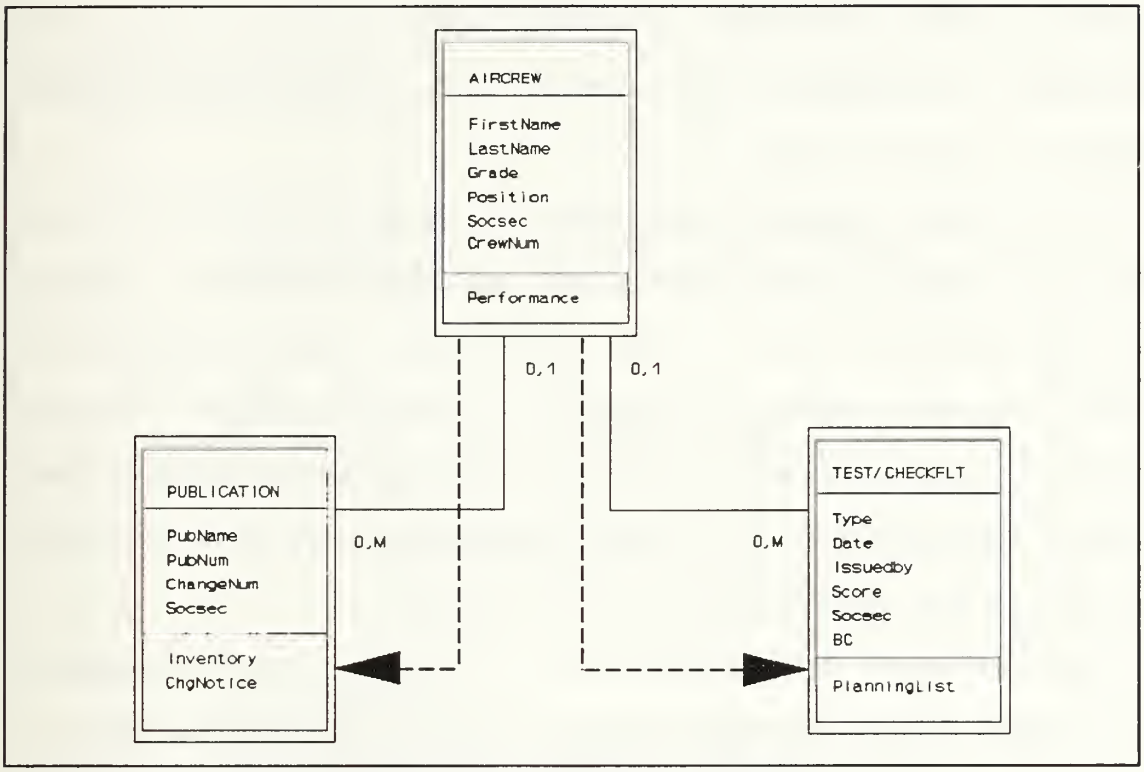

Figure 14 NATOPS Department Object Model

E. THE FLIGHT SCHEDULE DATABASE

The flight schedule database consists of data on flight events, simulator events, required events, staff crews, student crews, and aircrew added to or subtracted from an event. The data is group by the different components: flight event data includes event number, aircraft, preflight time,

take off time, land time, staff crew number and student crew number; simulator event data includes event number, simulator number, lab number, student crew number, staff crew number, brief time, and end time; required event data includes event number, event name, room number, start time, and end time; staff crew data includes staff crew number, and the various staff personnel assigned to that crew; student crew data includes the student crew number and the various student assigned to that crew.

The flight schedule database is used as a planning and execution tool to promulgate a daily flight schedule. Events are the primary focus of the database. These events are either ground training events, simulator events, flight events, or administrative events. Each event is given a time block. Additionally, the required assets and personnel are identified for each event.

The primary application is to ensure that required training is accomplished without double scheduling personnel. Assets may or may not be double scheduled depending on the event. In terms of assets, it is possible to double schedule most ground training events, but assets for flights and simulators cannot be double scheduled. Administrative events cannot be double scheduled unless they are of a large meeting type. An example would be an all officers meeting. This would apply to all officers not otherwise scheduled.

1. Flight Schedule Relational Diagram

The flight schedule relational diagram is depicted in Figure 15 and 16.

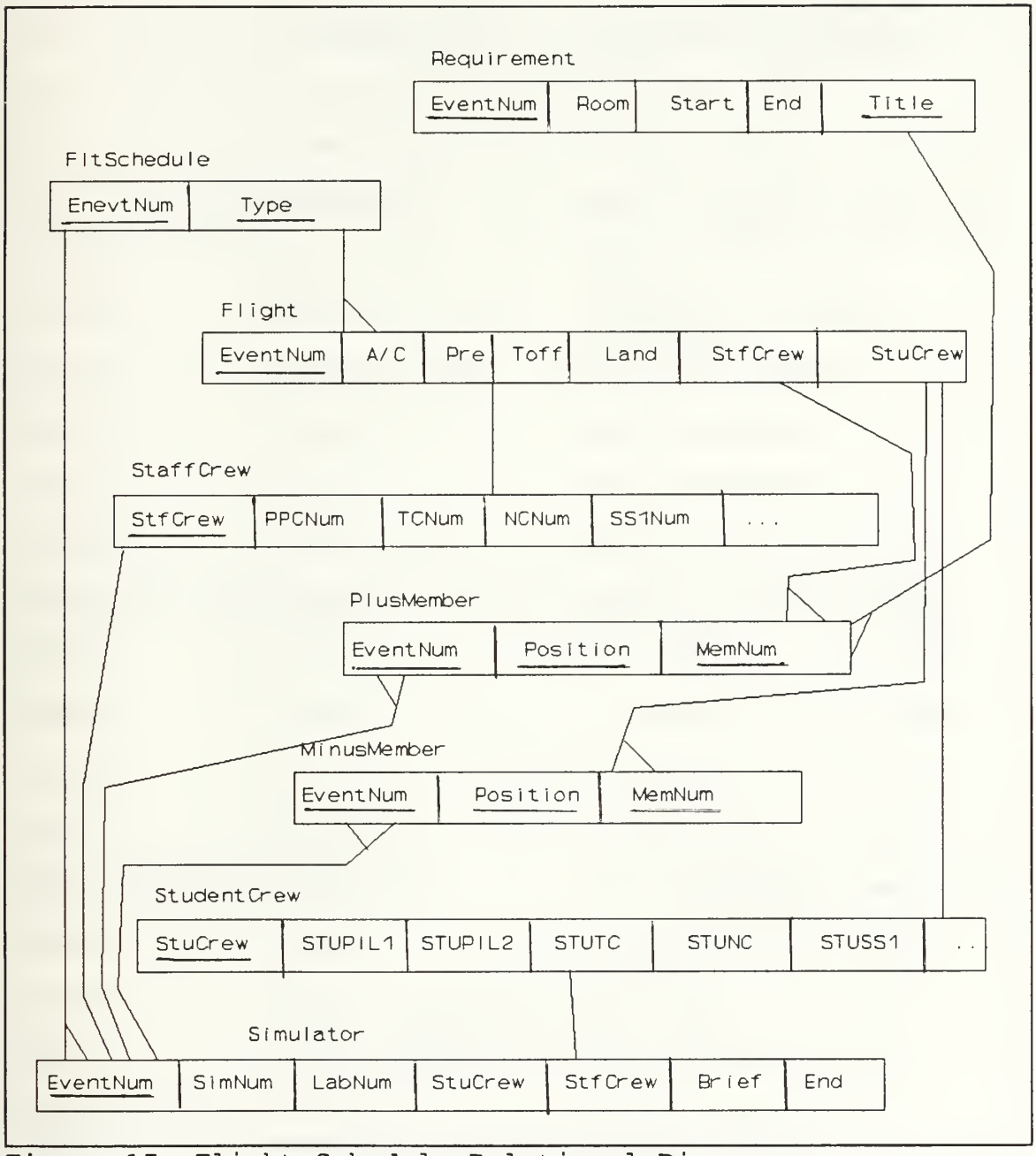

Figure 15 Flight Schedule Relational Diagram

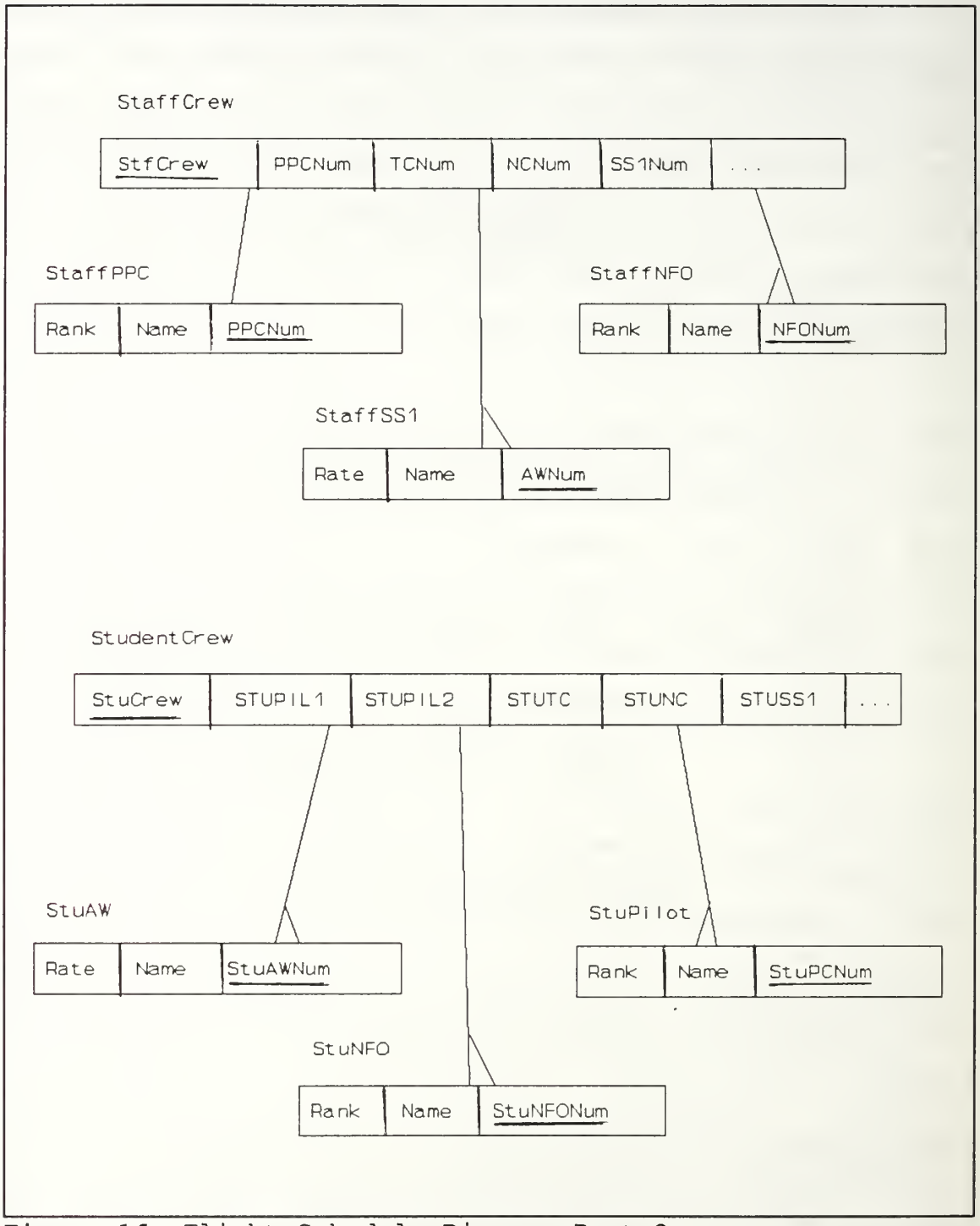

Figure 16 Flight Schedule Diagram Part 2

2. **Flight Schedule Data Dictionary**

NAME	TYPE(Length)	Description
EventNum	Numeric (2)	Event Number
Type	Character (6)	Flight or Sim
A/C	Character (5)	Aircraft Side Number
Pre	Numeric (4)	Preflight Time
Toff	Numeric (4)	Takeoff Time
Land	Numeric (4)	Land Time
StfCrew	Numeric(2)	Staff Crew Number
StuCrew	Numeric (2)	Student Crew Number
PPCNum	Numeric (3)	Pilot ID Number
TCNum	Numeric (3)	NFO ID Number
NCNum	Numeric (3)	NFO ID Number
SS1Num	Numeric (3)	AW ID Number
Position	Character (4)	Position Code (PPC,TC,..
MemNum	Numeric (3)	NFO,Pilot,... ID Number
STUPIL1	Numeric (3)	Student Pilot ID Number
STUPIL2	Numeric (3)	Student Pilot ID Number
STUTC	Numeric (3)	Student NFO ID Number
STUNC	Numeric (3)	Student NFO ID Number
STUSS1	Numeric (3)	Student AW ID Number
SimNum	Character (5)	Simulator ID Number
LabNum	Character (5)	Assigned Lab Number
Brief	Numeric (4)	Brief Time
End	Numeric (4)	Session End Time

Rank	Character (7)	Rank or Rate of Person
Name	Character (35)	Name of Person
StuAWNum	Numeric (3)	Enlisted AW ID Number
StuPCNum	Numeric (3)	Student Pilot ID Number
StuNFONum	Numeric (3)	Student NFO ID Number
Room	Character (4)	Room for Admin Event
Title	Character (5)	Name of Admin Event

3. Transformation Process

This is a more complex structure than the previous databases. To capture this complexity we use a number of whole-part structures and generalization-specialization structures. Starting with the fltschedule table, we transform this into an object with parts flight and simulator. Additionally, both staffcrew and studentcrew are treated as parts of simulator and flight. Likewise, plusmember and minusmember are treated as objects with connections to flight and simulator.

Some of the tables have similar attributes. To capture this, we use a generalization-specializations structure. We start by building a generalization class of identical attributes from the aircrew position related tables. We then add a number of specialization class-objects to cover the non-related attributes in staffppc, staffnfo, staffss1, stuaw, stupilot, and stunfo. This generalization-

specialization structure is also part of a whole structure to staffcrew or studentcrew.

To determine the methods, we must examine the applications of the database. The primary objective is to schedule required training without double scheduling personnel.

4. Flight Schedule Object Model

The flight schedule relational diagram is depicted in Figure 17 and 18.

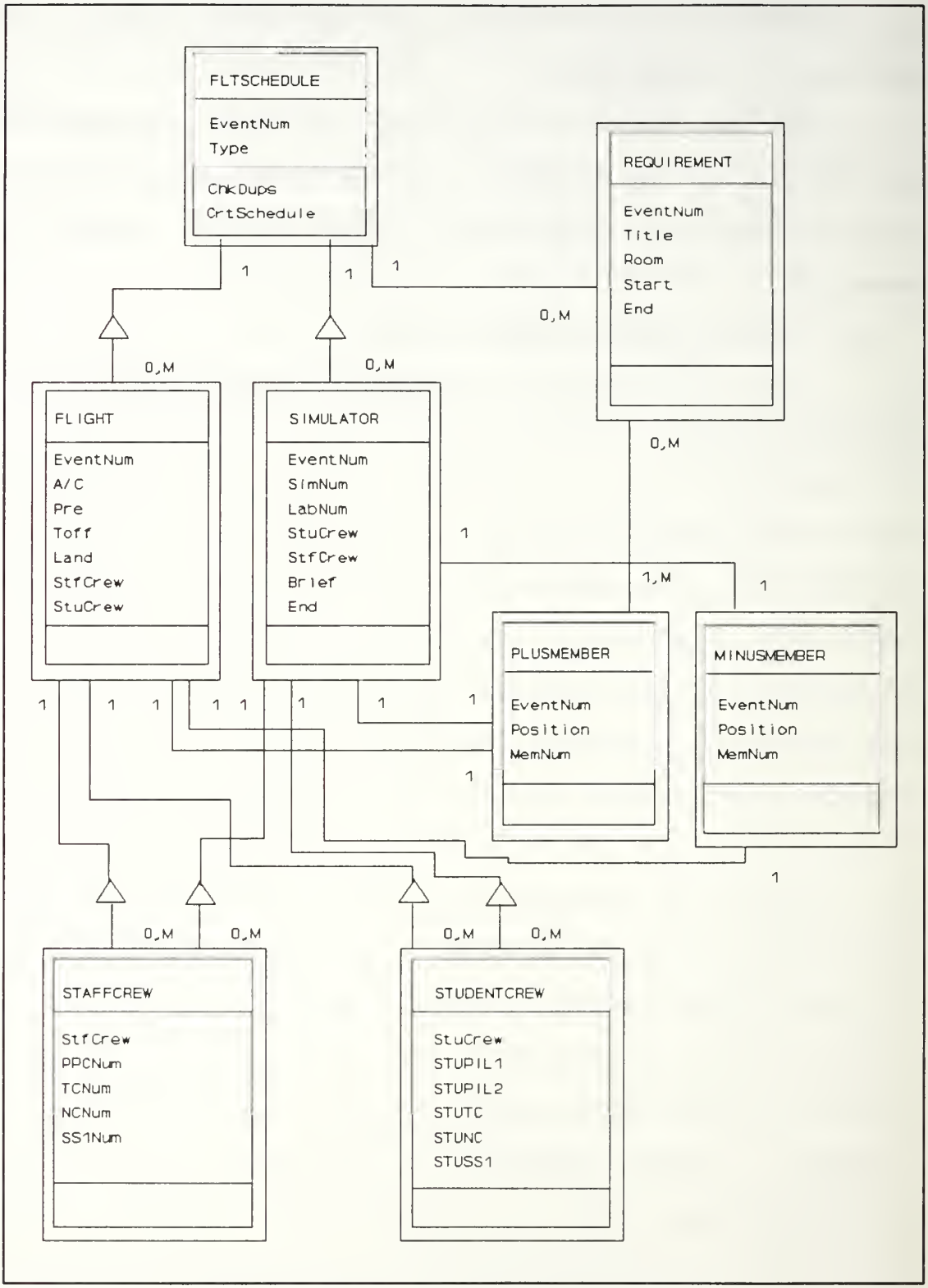

Figure 17 Flight Schedule Object Model Part 1

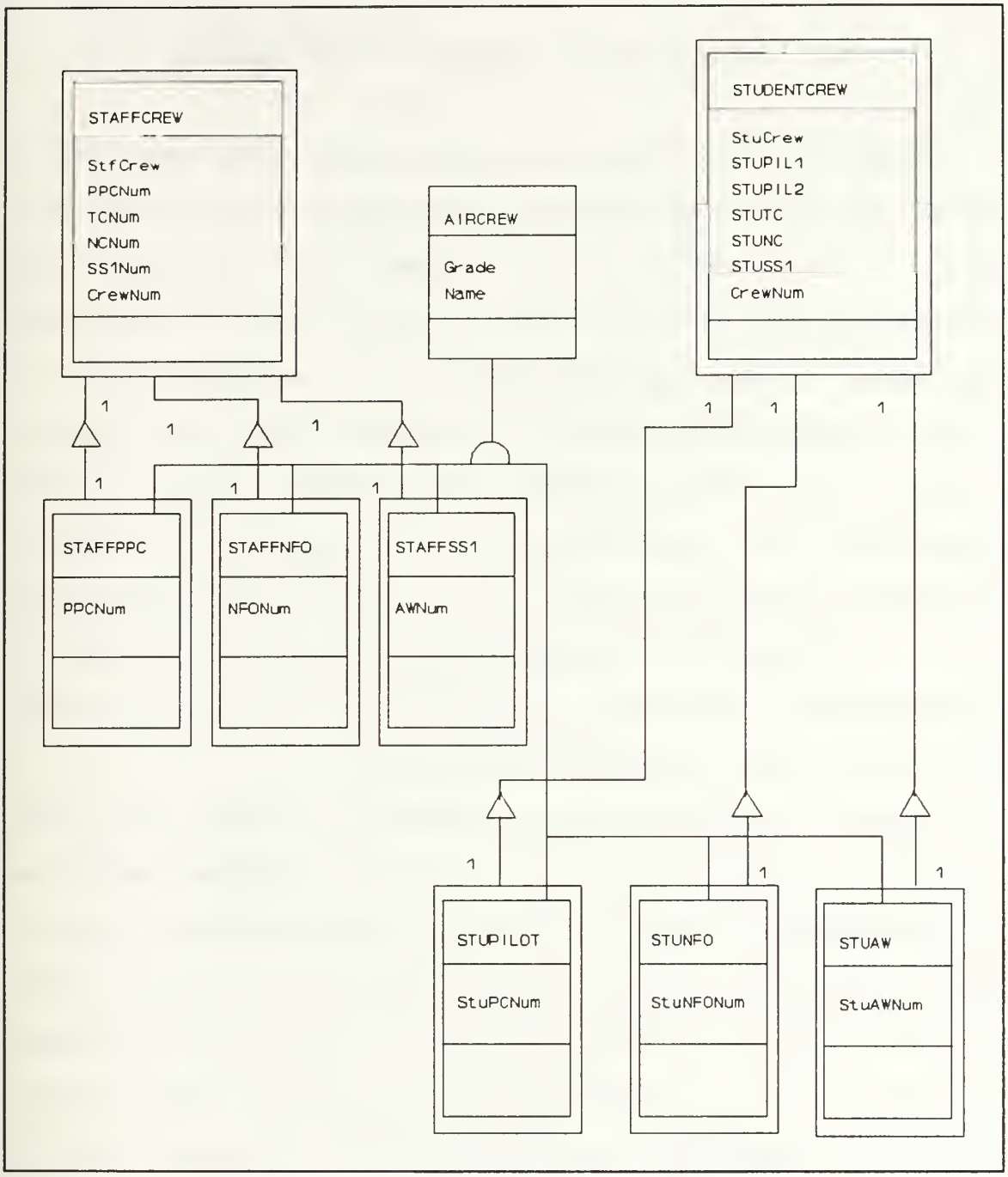

Figure 18 Flight Schedule Object Model Part 2

VI. FRAMEWORK FOR SEMANTIC HETEROGENEITY

Schematic and data conflicts between databases are a crucial problem in building multidatabase systems (Kim, 1991, pp.8). The conflicts are caused by structural and representational discrepancies or conflicts between component databases. To build a homogenizing layer, a global schema is needed. These conflicts must be resolved before constructing a global schema. To accomplish this we build a framework for enumerating and classifying the types of semantic heterogeneity that could exist in the object-oriented database model. The scenario developed in Chapter V will be used to illustrate the conflicts of the framework. The object model used is the model proposed in Chapter IV.

Semantic conflicts are divided into schema and data contents conflicts (Kim, 1991, pp.12-18). Schema conflicts are classified broadly by the level at which they could occur. These levels are: object level conflicts, attribute level conflicts, object-attribute level conflicts, and object method conflicts. The framework covers two primary types of data conflicts; inconsistencies and different representations for the same data. Each level of the framework is discussed in detail. Figure 19 is an overview of the types of conflicts that are described.

A. Schema Conflicts

1. Object Level Conflicts

- Object name conflicts

- Object structure conflicts

2. Attribute Level Conflicts

- Attribute name conflicts

- Attribute constraint conflicts

- Attribute structure conflicts

3. Object-Attribute Level Conflicts

- Object-attribute structure conflicts

4. Method Conflicts

- Method name conflicts

- Method connection conflicts

B. Data Conflicts

1. Inconsistencies

2. Different Representations for the Same Data

- Different expresions

- Different units

- Different granularities

Figure 19 Object Model Semantic Conflicts

A. SCHEMA CONFLICTS

1. Object level conflicts

Object level conflicts occur when the heterogenous databases use different representations for similar objects. These can be decomposed into object name conflicts, or object structure conflicts.

a. Object name conflicts

Object name conflicts are of two types. The first is a homonym problem exhibited when the same name is used in two databases to denote semantically different objects. The second is a synonym problem that occurs when the same name is used to denote semantically different objects. The database scenario exhibits both.

The homonym conflict is seen in the following example. The Flight Schedule model has an object called requirement that refers to a required administrative event. The Flight Physiology object-model has an object called requirement that refers to required aircrew qualification. Though these object have the same name they are not semantically related. Figure 20 illustrates the problem.

The synonym conflict is seen in the following example. The classified library object-model has an object call docholder that refers to a person who has custody of a document. The NATOPS Department object-model has an object

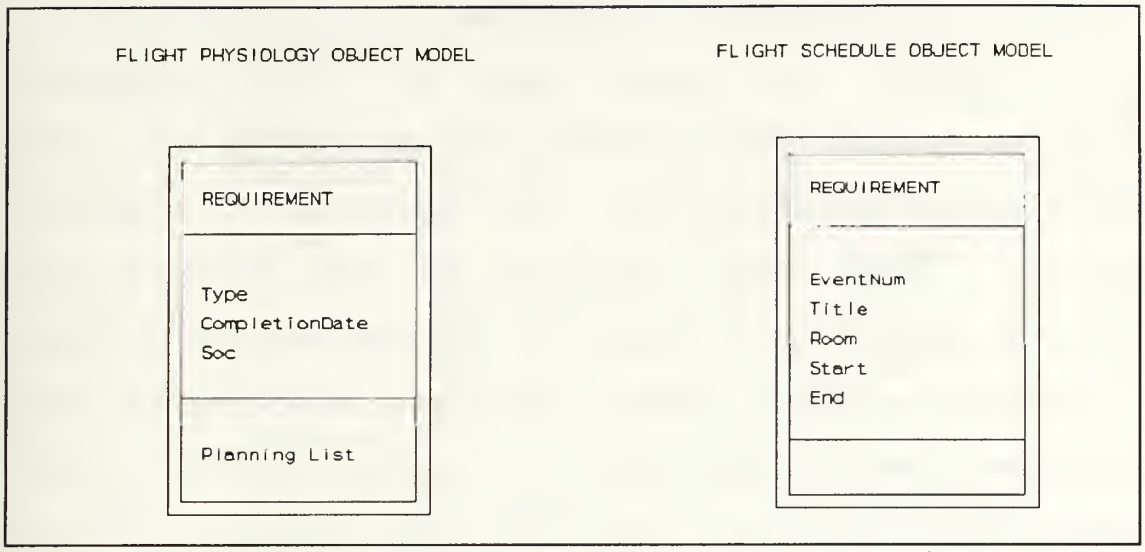

Figure 20 Example of Homonym Object Name Conflict

called aircrew that refers to a person who also has custody of a document. These two objects are semantically equivalent and represent a person who has custody of a document, yet they have different names. Figure 21 illustrates the problem.

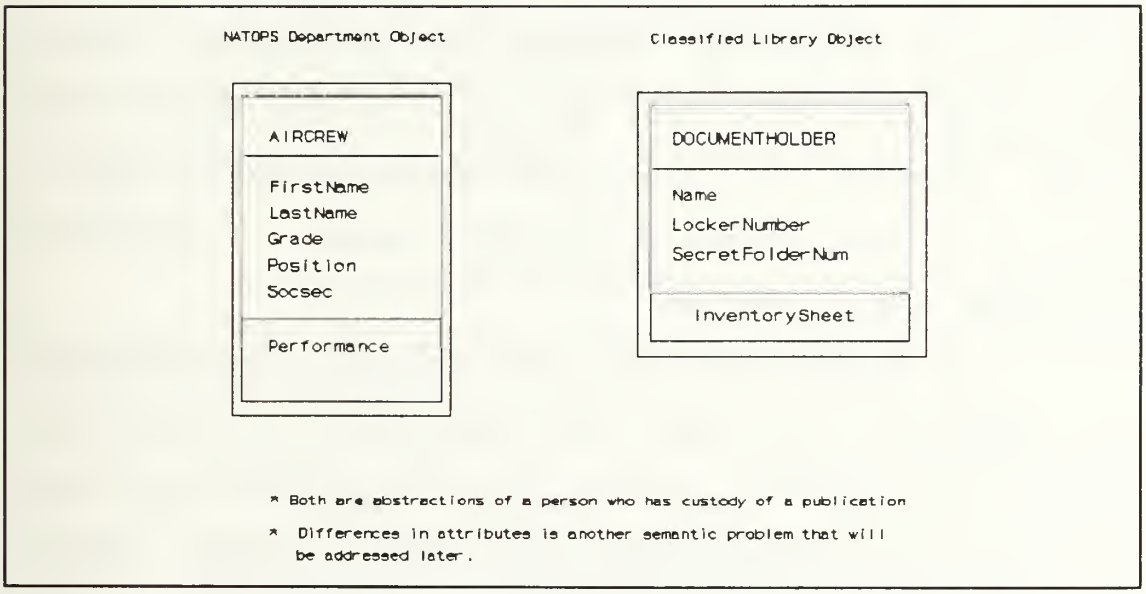

Figure 21 Example of Synonym Object Name Conflict

b. Object structure conflicts

There are three types of object structure conflicts; generalization-specialization, whole-part, and generalization-specialization to whole-part structure conflicts. With these conflicts one must focus on the attributes required in the query or application precipitating the conflict. Object method conflicts are treated as a special case and addressed later.In the generalization-specialization structure conflicts, the attributes of an object in one model are contained in a generalization-specialization structure of another model. Consider the name, rank, and social security number attributes. In the NATOPS department model this information is contained in the aircrew object. In the flight physiology model this information is contained in the servicemember-officer generalization-specialization structure. However, in the aircrew object, grade encompasses both rate and rank in the officer and enlisted objects of the generalization-specialization structure in the flight physiology model. Figure 22 illustrates this situation.

In a whole-part structure conflict, the attributes of an object in one model are contained in a whole-part structure of another model. Consider the attributes name and crew number. This information is contained in the aircrew object in the NATOPS model, and in the whole-part structure of staffcrew object and the studentcrew object in the flight

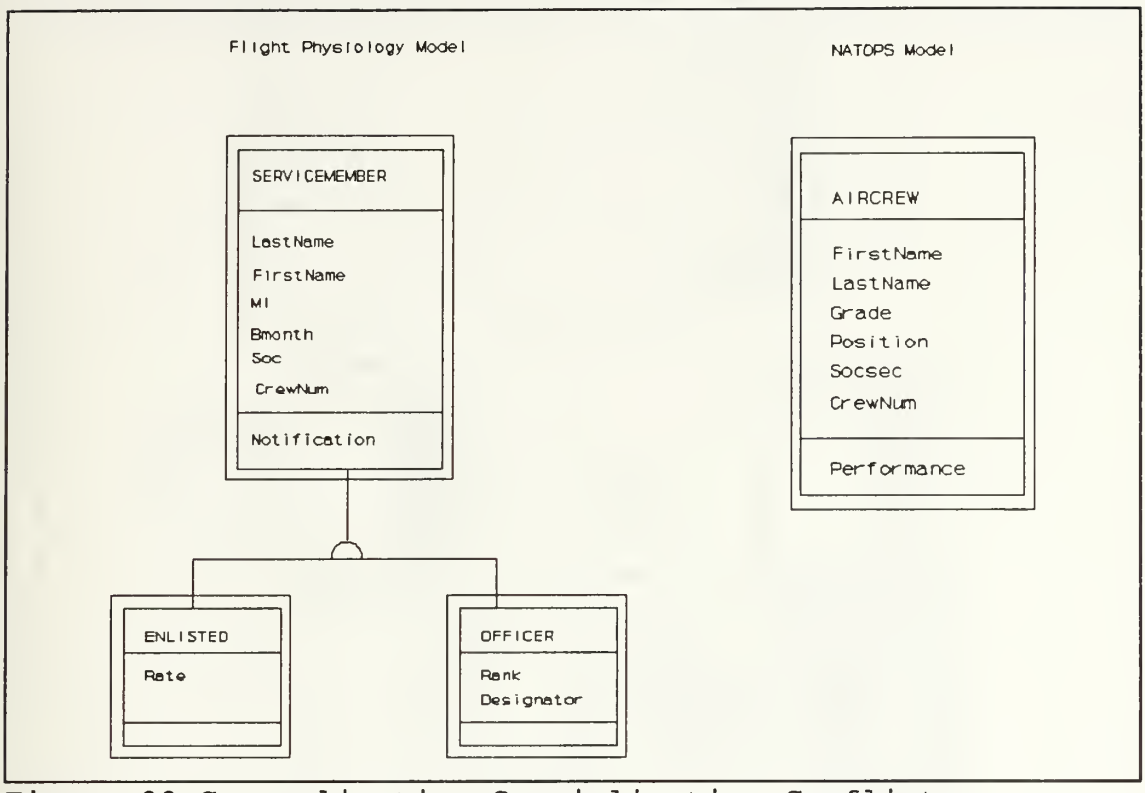

Figure 22 Generalization-Specialization Conflicts

schedule object model (The generalization-specialization structure aircrew-staffPPC, aircrew-staffNFO, etc., does not cause an additional conflict due to the concept of inheritance of the generalization objects). Figure 23 illustrates this situation.

In the generalization-specialization to whole-part structure conflict the attributes of interest are contained in a generalization-specialization structure in one model, and a whole-part structure of another model. Consider the attributes of name, grade, and crew number where grade is either the rank of an officer, or the rate of an enlisted personnel. This information is contained the generalization-specialization

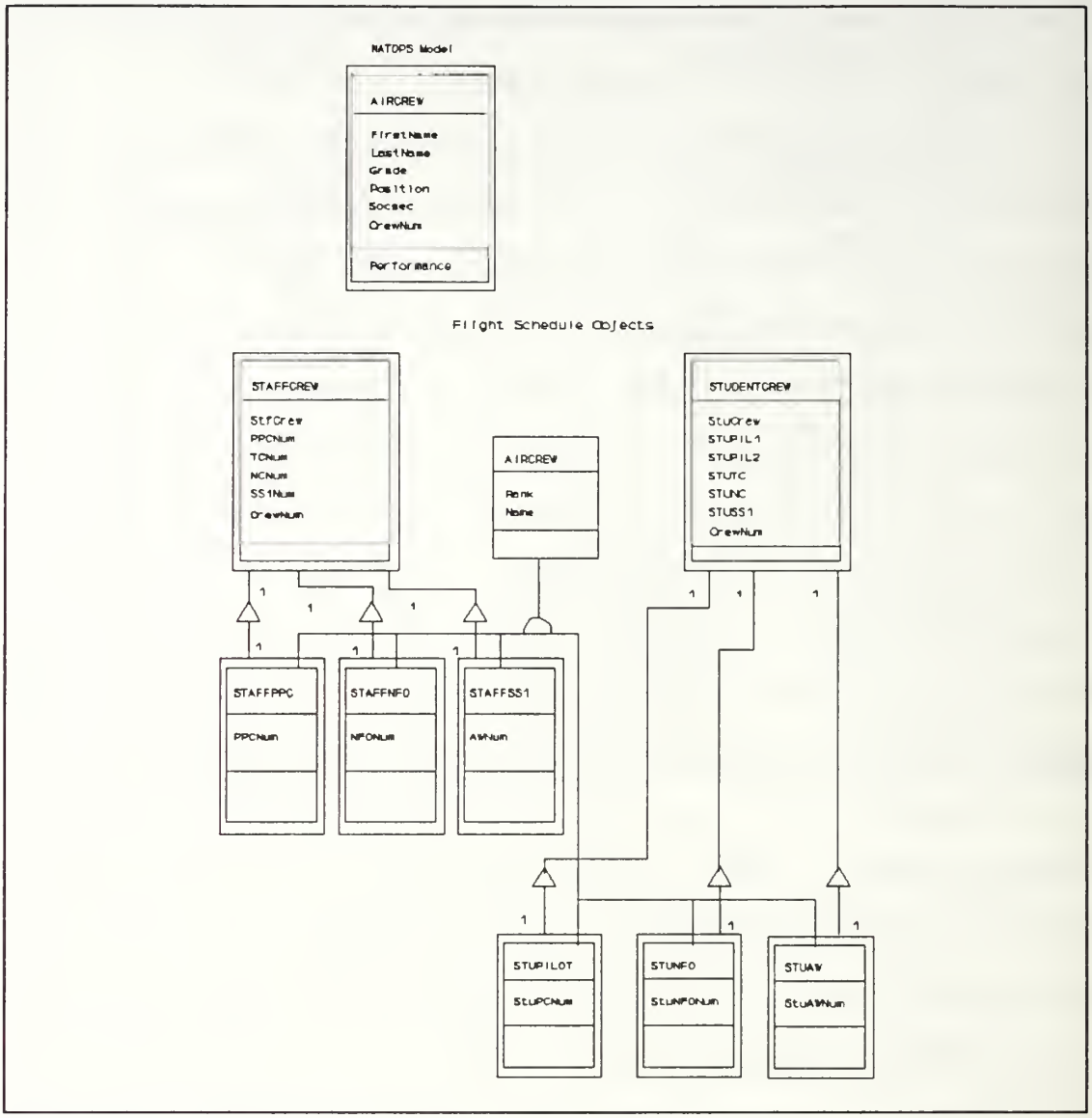

Figure 23 Whole-Part Structure Conflicts

structure in the flight physiology model and in a whole-part structure in the flight schedule model. Figure 24 illustrates this situation.

In the generalization-specialization to generalization-specialization conflict, the attribute of

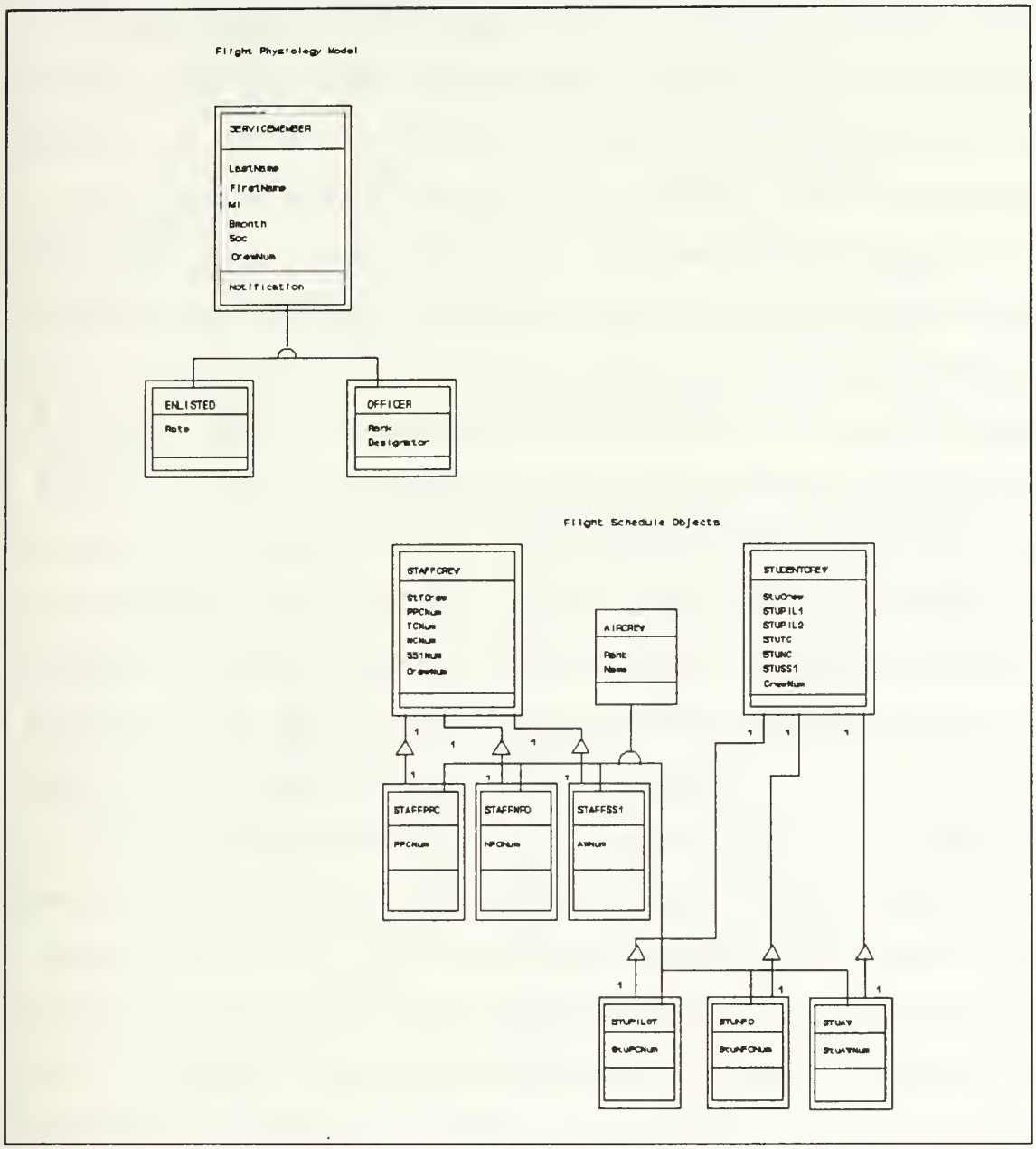

Figure 24 Generalization-Specialization to Whole-Part Structure Conflicts

interest in one generalization-specialization structure are found in a different generalization-specialization structure of another object model. Consider the attributes of grade and name, were grade is either an officer's rank or an enlisted's

rate. This information is found in a generalization-specialization structure in the flight schedule model as well as a generalization-specialization structure in the flight physiology model. Figure 25 illustrates this situation.

In the whole-part to whole-part conflict, the attribute of interest in one whole-part structure are found in a different whole-part structure of another object model. Our example does not contain an example of this conflict. Modifying the NATOPS model so that publication is now a whole-part structure. The natlib object will contain the name of the +NATOPS library (assume we can now have more than one) and it has NATOPS position publications (natpub) and crew station maintenance manuals (crewman) as parts. The attributes of interest are the library name and all the publications contained in the libraries. This modified model contains a whole object natlib that contains the attribute LibName and parts natpub and crewman that contain all the publications. The classified library model has a whole object called library and two parts. One part is publication which contains all the publication names in the library and the others are abstracts of document holders. Figure 26 illustrates the modification and the conflict situation.

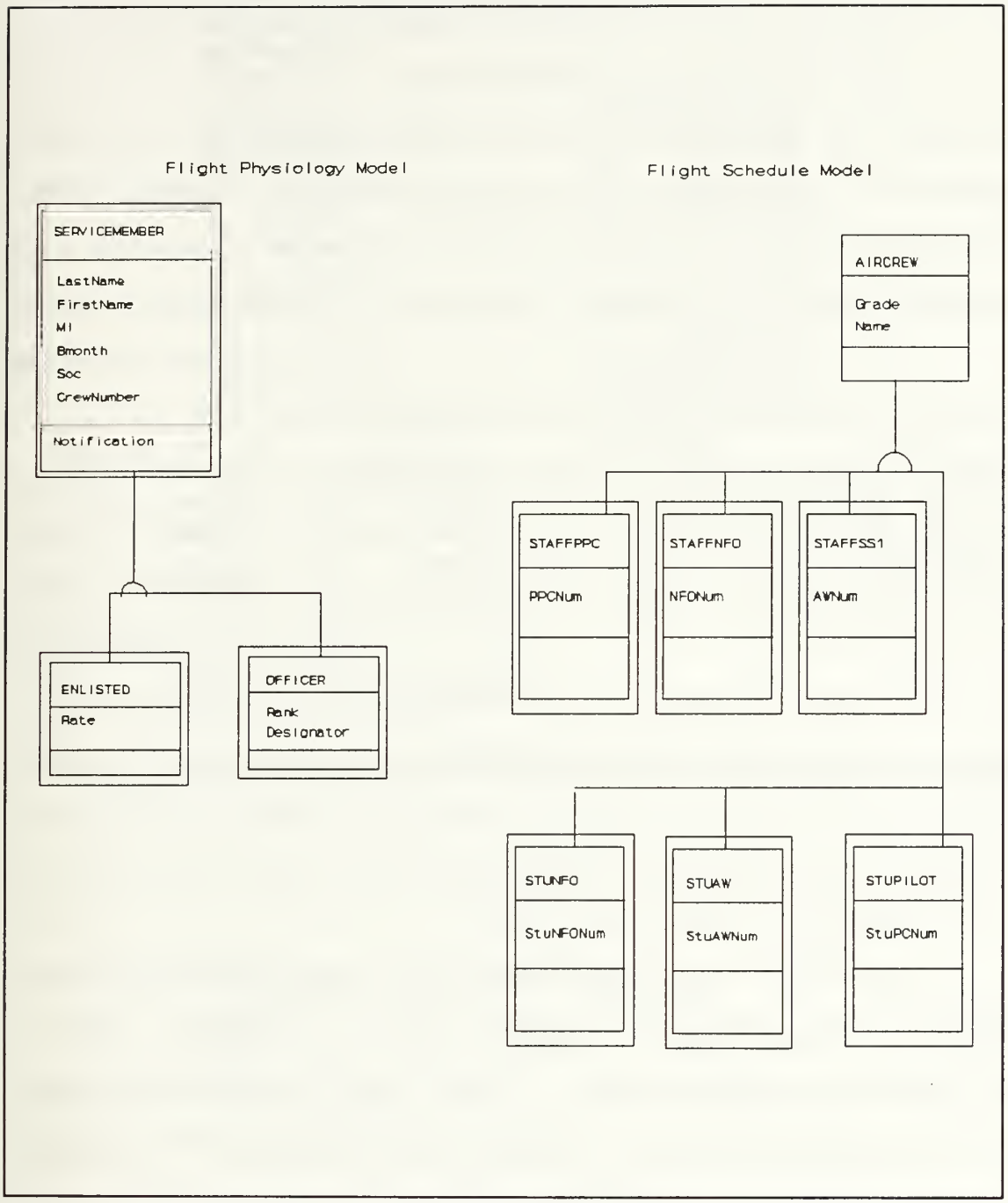

Figure 25 Generalization-Specialization to Generalization-Specializtion Strucure Conflicts

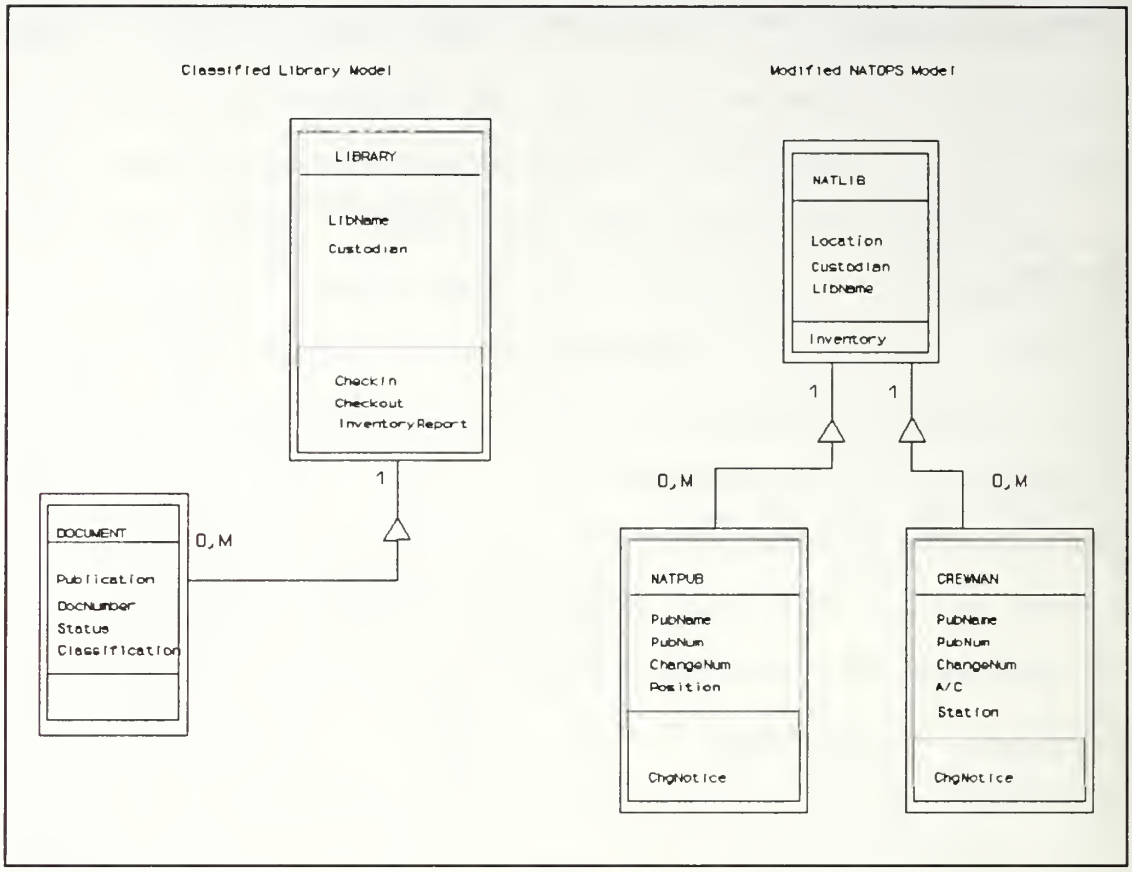

Figure 26 Whole-Part to Whole-Part Structure Conflicts

2. Attribute Level Conflicts

Attribute level conflicts occur when the heterogenous databases use different delineations to represent similar attributes of abstractions. These can be decomposed into attribute name conflicts, attribute constraint conflicts or attribute structure conflicts.

a. Attribute name conflicts

Attribute name conflicts are of two types. The first is a homonym problem exhibited when the same name is used to denote semantically different attributes. The second is a synonym problem that occurs when the same name is used to denote semantically different attributes. The database scenario exhibits both.

The synonym conflict is exhibited in the following example. In the classified library model, the attribute 'name' refers to the name of a person. In the flight physiology model, 'name' refers to the name of a requirement not a person.

The homonym conflict is seen in the following example. In the flight physiology model the attribute name for a social security number is 'soc'. In the NATOPS model the attribute name for a social security number is 'Socsec'.

b. Attribute constraint conflicts

There are two types of attribute constraint conflicts, data type and attribute integrity-constraint conflicts. The data type conflicts occur when semantically equivalent attributes in different models have different data types or data length. In our example the attribute that represents a social security number in the classified library model, named 'SS', is of type numeric. In the flight

physiology model the same attribute, named 'soc', is of type character. Similarly, in one model the length of the social security field could be 9, while in another it could be 11 to accommodate two hyphens. An example is "045-62-3436" vice "045623436".

The attribute integrity-constraint conflict occur due to dissimilar definitions of attribute constraints of similar attributes in the different models. In our example, in the flight physiology model the allowed values of rank are Ens, Ltjg, LT, LCDR, CDR, CAPT, RADM, and VADM. In the flight schedule model, the allowed values are Ens, Ltjg, LT, LCDR, CDR, and CAPT.

c. Attribute structure conflicts

Attribute structure conflicts occur when a group of attributes in one model are semantically equivalent to a single or lessor number of attributes in another model. These occur when semantically similar objects have a different number of attributes. This can be further decomposed into a missing attributes conflict, or missing but implicit attribute In the missing attribute conflict one object is missing attributes that a semantically equivalent object contains. The attribute is truly missing and cannot be deduced. In our example, the flight physiology model has an object called requirement. The attributes are type, completiondate, and

soc. The NATOPS department model object test/checkflt has type, date, issuedby, socsec, score, and bc as attributes. The objects are semantically similar. Requirement.type is equal to test/checkflt.type. Requirement.completiondate is equivalent test/checkflt.date. Requirement.soc is equivalent to test/checkflt.socsec. Test/checkflt has the additional attributes of issuedby, score, and bc. None of these can be deduced in the requirement object.

In the missing but implicit attribute conflict, attributes in one object are missing, but can be deduced. This can be a subtle distinction. Our scenario does not have a good example of this. To illustrate we will change the last example slightly. The requirement object will remain the same. The test/checkflt object will now have the attributes type, date, socsec, and name. Name refers to a persons name that is associated with the socsec (social security number). Now the missing attribute of name can be deduced in the requirement object.

3. Object-Attribute Level Conflicts

Object-attribute level conflicts occur when information in one model is reflected by an attribute and by an object in another. The database scenario does not have a good example of this. For our purposes assume the NATOPS model has an object called aircraft. The object aircraft contains attributes side_number, type, and version. The

flight schedule model has an attribute called aircraft that identifies the aircraft by side number. The implication is that knowing the side number implies type and version. So, the information in an attribute in one model is contained in the object of another model.

4. Method Conflicts

Method conflicts fall into two general types. The classes are divided by conflicts that concern methods unique to one model, or conflicts that concern the global or integrated model. The types of method conflicts exhibited are dependant on the type of homogenizing strategy employed.

a. Method name conflicts

With methods unique to one model, the primary conflict that arises in a global view is a method name conflict. This occurs when two heterogeneous models contain methods with the same name and the method is being employed while exploiting a global schema. In our example, the flight physiology model has an method called planning list that works in conjunction with the attributes of the requirement, enlisted, and officer objects of that model. The NATOPS department model also has a method called planning list. It was designed to work with the objects of that model. So, in an actual or virtual global schema, a conflict would occur if both methods are transported to the global schema.

b. Method message and instance connection conflicts

The second class of method conflicts occurs when a method in a heterogeneous model is extended to apply to other heterogeneous models in a real or virtual global schema. Again the possibility of a naming conflict exists. However, the conflicts here would more likely be related to conflicts of message connections and/or instance connections. The message and instance connections would relate to attributes that exist in the local model or view. Extending the method to the global model or global schema would entail establishing message or instance connections to other heterogeneous models. These connections could cause the manifestation of all previously mentioned schematic conflicts.

B. DATA CONFLICTS

Data conflicts are of two distinct types; inconsistencies, or different representations for same data. Data conflicts are independent of the schema involved.

1. Inconsistencies

Inconsistencies are generally due to failures in maintaining a database, such as failing to keep the database up to date and failure to enforce integrity rules (Kim, 1991, pp.17). The problems with inconsistencies can be expressed as data entry errors or obsolete data.

Data entry errors occur when equivalent attributes in different object models, which are expected to have the same

value, have different values. In our example the attribute rank appears in the flight schedule and flight physiology models. Rank is part of the same abstraction. It naturally follows that an instance of similar objects in the two different models should have the same rank. If however in one data base, the rank of John Smith is LT, and in the other the rank of the same instance is Ltjg, we have a case of wrong data in one of the models. If this was due to an entry error, it would be classified as an incorrect-data entry. It naturally follows that an instance of similar objects in the two different models should have the same rank. If however, in one data base, the rank of John Smith is LT, and in the other the rank of the same instance is Ltjg, we have a case of wrong data in one of the models. If John Smith was recently promoted to the rank of LT, and this was updated in one model and not the other, this would be a case of obsolete data.

2. Different Representations for the Same Data

The three aspects of data that lead to its representation are expressions, units, and granularity. These are the areas of representational conflict we will examine further.

a. Different expressions

Conflicts in expression can occur when two models use the same data, but express it differently. In our example the data in rank can be expressed as Ens, Ltjg, LT, LCDR, CDR,

or CAPT. This data could also be expressed as 0-1, 0-2, 0-3, 0-4, 0-5, or 0-6. In the U.S. Navy military rank structure these codes are different expressions for the same data. Using the same example, ensign, lieutenant junior grade, lieutenant, etc., could be spelled out instead of using the abbreviations. This would be a case of using different words or strings for the same data.

b. Different unit for the same data

These conflicts arise when two models use different units for similar numeric data. In our example we could included an attribute qual_duration of type numeric to both the NATOPS department model and the flight physiology model. In one we could have the numeric represent months, while in the other the numeric represent years. So, even if both attributes hold the same value they represent different things.

c. Different granularity

Conflicts in granularity occur when two models use values from the domain of different cardinalities for the same data (Kim, 1991, pp.17). For example in our scenario the NATOPS model has an attribute score. The data type is a numeric from range 0.0 to 4.0 reflecting a 4.0 grading scale. We can added a semantically equivalent attribute to the flight physiology model and make it of an enumerated data type of fail, very poor, poor, satisfactory, good, very good, and

outstanding. The domains now represent the same data, but use different granularity.

C. CONCLUSION

In this chapter we attempted to develop a complete framework for enumerating and classifying schematic and data conflicts in a object-oriented database model. In the next chapter several ideas are proposed to resolve these conflicts.

VII. PROPOSED SOLUTIONS TO SCHEMATIC AND DATA CONFLICTS

In this chapter we will consider the problems encountered and the feasible solutions for querying the global schema. We will focus on integrating the classified library, and the NATOPS department databases. Problems encountered when adding, deleting, or modifying data in a global schema are not addressed.

To allow for queries to span these two databases a federated approach is used. As indicated earlier the first step in this approach is to transform the component database schemas into equivalent schemas in the object-oriented model. This was accomplished in Chapter IV. The second step is to examine the component databases in the object-oriented model and integrate them into a global schema after identifying and resolving the schematic conflicts.

A. SCHEMA INTEGRATION RESOLUTION

The first step in this process is conflict identification. To aid in identifying the schema and data conflicts we rely on the framework developed in Chapter VI.

1. Object Level Conflict Resolutions

a. Object structure conflict resolutions

We begin by examining the library and NATOPS department object models for object structure conflicts. The

classified library model, shown in Figure 28, has a generalization-specialization structure made up of a generalization object 'documentholder' and specialization objects 'studentholder' and 'staffholder'. This structure is semantically equivalent to the NATOPS object 'aircrew', shown in Figure 27. This is the only object structure conflict present, and is a generalization-specialization conflict. To resolve structure conflicts in preparation for integration, we remap the structure of the simpler model to a more complex one to better match the structure of the complex model.

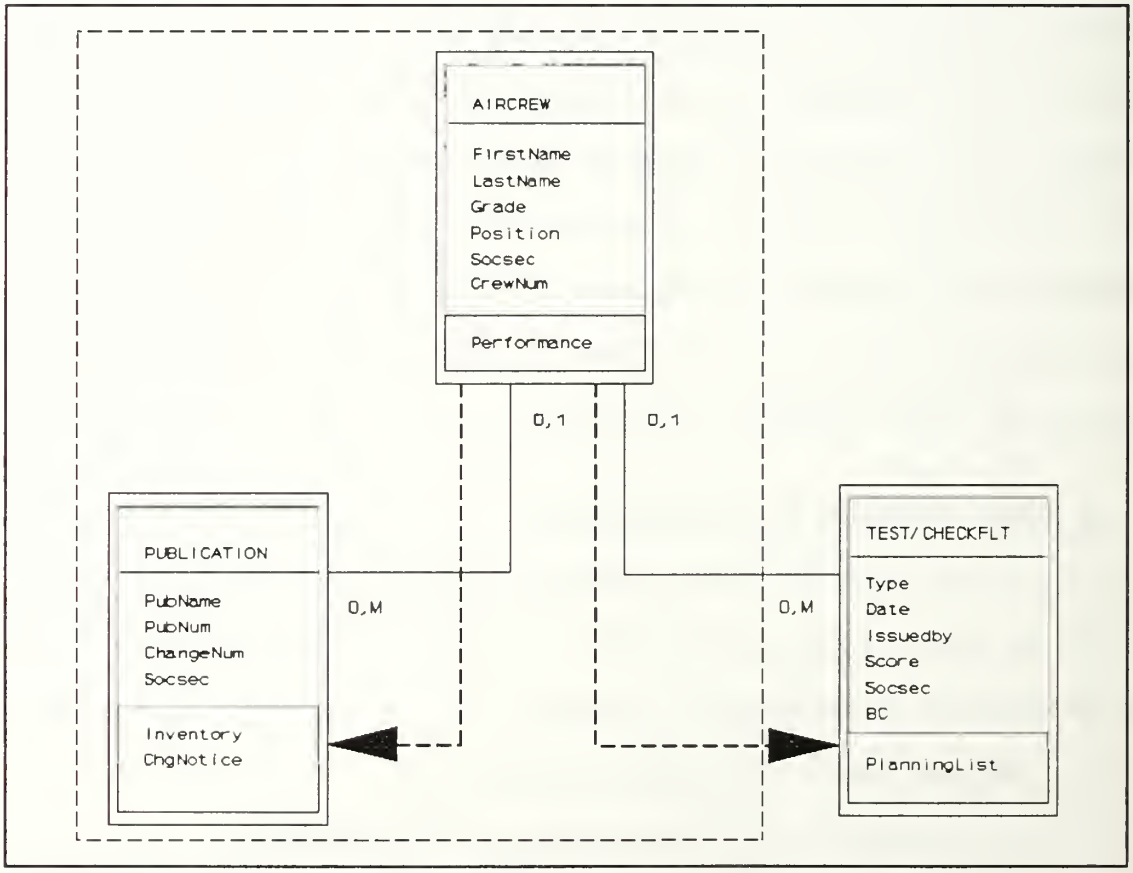

Figure 27 NATOPS Department Object Model

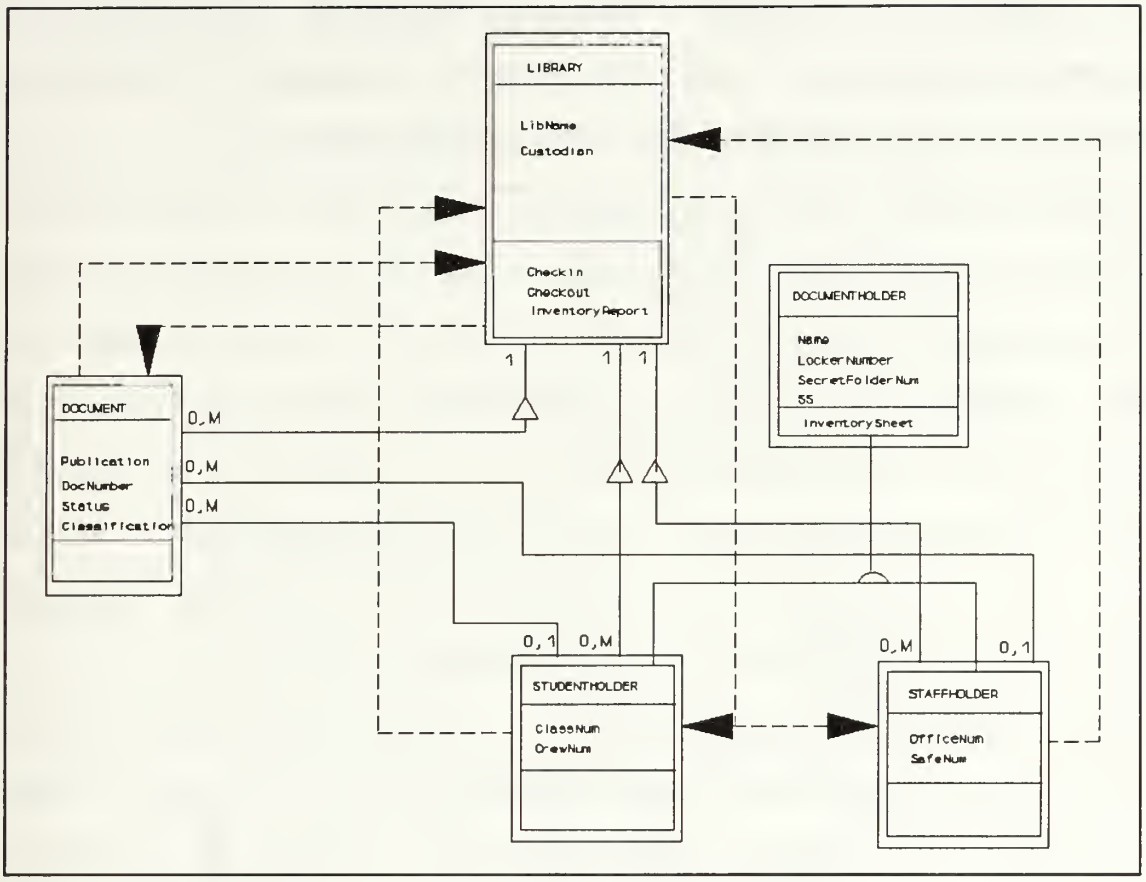

Figure 28 Classified Library Object Model

When examining the two structures, it is obvious that the classified library model is the more complex model. To remap the NATOPS structure, we rely on the attribute 'position' to distinguish between student and staff personnel. This attribute starts with 'stu' for student aircrew. For example, a staff pilot is entered as pilot for position in the NATOPS database while a student pilot is entered as stupilot. To develop the structure we use the aircrew object as a generalization object, and add staffaircrew and stuaircrew as specialization objects. Position and crewnumber attributes are moved to the corresponding specialization objects. Once

the models are remapped into an equivalent structure, the object structure conflicts are resolved. Figure 29 illustrates the NATOPS data model remapping.

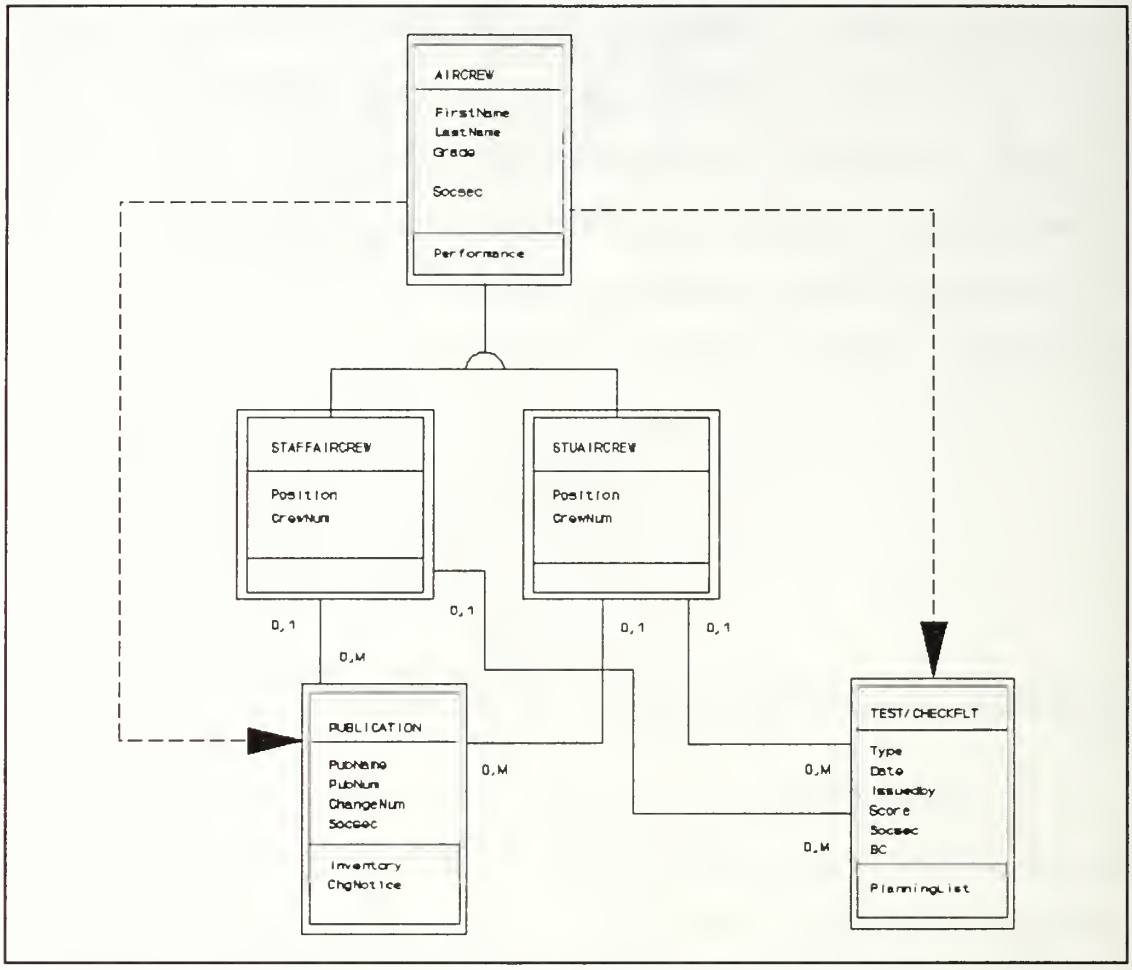

Figure 29 NATOPS Department Remapped

b. Object name conflict resolutions

The next conflicts to resolve are the object name conflicts. When resolving these conflicts we focus on the library model and the remapped NATOPS model. The following object name conflicts are present: Library.documentholder is equivalent to NATOPS.aircrew, Library.studentholder is

equivalent to NATOPS.stuaircrew, Library.staffholder is equivalent to NATOPS.staffaircrew, and Library.document is equivalent to NATOPS.publication. These object name conflicts are resolved in the global schema by using a global object that corresponds to each equivalent pair. The mapping to/from the local objects is handled by a look-up table accessible by the global controller. Information in the look-up tables are accessed at run-time by the global controller to direct queries to the component databases. The global controller is explained in detail in section C.

For our example, Library.documentholder and NATOPS.aircrew correspond to a global generalization object named holder. Library.studentholder and NATOPS.stuaircrew correspond to a global specialization object called student. Library.staffholder and NATOPS.staffaircrew correspond to a global specialization object called staff. Library.document and NATOPS.publication correspond to a global object called document. The Library.library and the NATOPS.test/checkflt objects have no semantic equivalents. Thus, they each become global objects.

2. Attribute Level Conflict Resolutions

With the object level conflicts resolved, we turn our attention to the attribute level conflicts. To determine what attribute level conflicts are present, we examine the

attributes of the semantically equivalent objects. Again, we use the conflict framework developed in Chapter IV as a guide.

a. Attribute name conflict resolutions

The first apparent problems are attribute name conflicts. Library.documentholder has an attribute called 'ss' and NATOPS.aircrew has an attribute called 'socsec'. Both refer to a social security number. The attributes 'publication' in Library.document and 'pubname' in NATOPS.document are semantically equivalent, and 'Docnumber' in Library.document and 'pubnum' in NATOPS.publication are also semantically equivalent. These depict the synonyms conflict. They are resolved in the global schema by using a global object attribute that corresponds to each equivalent pair. The mapping to/from the local object attributes is handled by a look-up table accessible by the global controller.

b. Attribute constraint conflict resolutions

The Library attribute 'ss' is defined as a nine digit numeric type, and the NATOPS attribute 'socsec' is defined as a nine place character type. The NATOPS 'crewnum' is defined as a four place character while the Library 'crewnum' is defined as a two place character. The attributes 'publication' in Library.document and 'pubname' in NATOPS.document are semantically equivalent, and 'pubnum' in publication and 'docnumber' in document are semantically

equivalent. The types of these equivalent attributes are the same, but the length are different. These are all attribute constraint conflicts. Again, the resolution of this at the query level is provided by a look-up table accessible to the global controller.

c. Attribute structure conflict resolutions

The 'name' attribute in Library.documentholder is semantically equivalent to 'grade' plus 'firstname' plus 'lastname' in NATOPS.aircrew. This is an attribute structure conflict where a group of attributes in one model are semantically equivalent to a single attribute in another model. The resolution of this conflict is accomplished at two levels. First, at the global schema level, an object corresponding to this pair will contain the more detailed attribute structure (i.e., grade, lastname, and firstname). Second, the global controller uses a look-up table to resolve decompose and translate a query to the global schema into subqueries to the corresponding data models schemas. An example of an element in a look-up table to resolve this conflict is as follows: Library.name = NATOPS.grade + NATOPS.firstname + NATOPS.lastname.

Additionally, the attributes 'lockernumber' and 'secretfoldernum' in Library.documentholder are not semantically contained in NATOPS.aircrew. Library.studentholder has an additional attribute 'classnum'

and NATOPS.stuaircrew has an additional attribute 'position'. They are, however, semantically unrelated. Library.staffholder and NATOPS.Staffaircrew are both specialization objects of the corresponding Library.documentholder and NATOPS.aircrew generalization pair. None of the specialization attributes in these specialization objects correspond to each other. These are all missing attribute conflicts. To resolve these, the attributes in the global schema represent the union of the attributes in the local schemas.

3. Object-Attribute Level Conflict Resolutions

Our example does not contain any object-attribute level conflicts. These conflicts are present when an attribute in one model corresponds to an object in another model. At the global schema level, the solution is to transform the attribute into an object. This is similar to the structure remapping presented earlier, where by the structure of one model is remapped so both models, that will be integrated, have similar objects. At run-time, the global controller uses look up tables to resolve the conflict between the global schema object and the local schema.

4. Method Conflict Resolutions

The method conflict we are concerned with, are the conflicts that arise when local methods are extended to the global schema. Local methods are designed to act on attributes of local objects. To use these methods in a global

view, the appropriate message connections must be mapped to the appropriate global objects. The resolution of this conflict is dependant on the resolution of all other conflicts. To simplify this exercise, we are building a global schema without the intention of extending local methods to the global schema.

B. CONSTRUCTING THE GLOBAL SCHEMA

1. The Global Objects

The first two candidate global objects are the non-equivalent objects. These are the Library.library and the NATOPS.test/checkflt objects. In this case, for simplicity the global objects maintain the same names and attributes.

The rest of the global objects come from the semantically equivalent pairs. From the Library.documentholder and NATOPS.aircrew pair we build a generalization object called 'holder' and include the attributes grade, firstname, lastname, lockernumber, secretfoldernum, and socsec. From the Library.studentholder and NATOPS.stuaircrew pair we build a specialization object called 'student'. This object includes the attributes crewnum, classnum, and position. The Library.staffholder and NATOPS.staffaircrew pair yields the global specialization object 'staff' with the attributes position, crewnum, safenum, and officenum. The final global object comes from the Library.document and NATOPS.publication pair. We name this object 'document' and give it the

attributes name, number, changenum, status, classification, and socsec.

2. The Global Schema Structure

Figure 30 depicts the global schema when the NATOPS and Library models are combined.

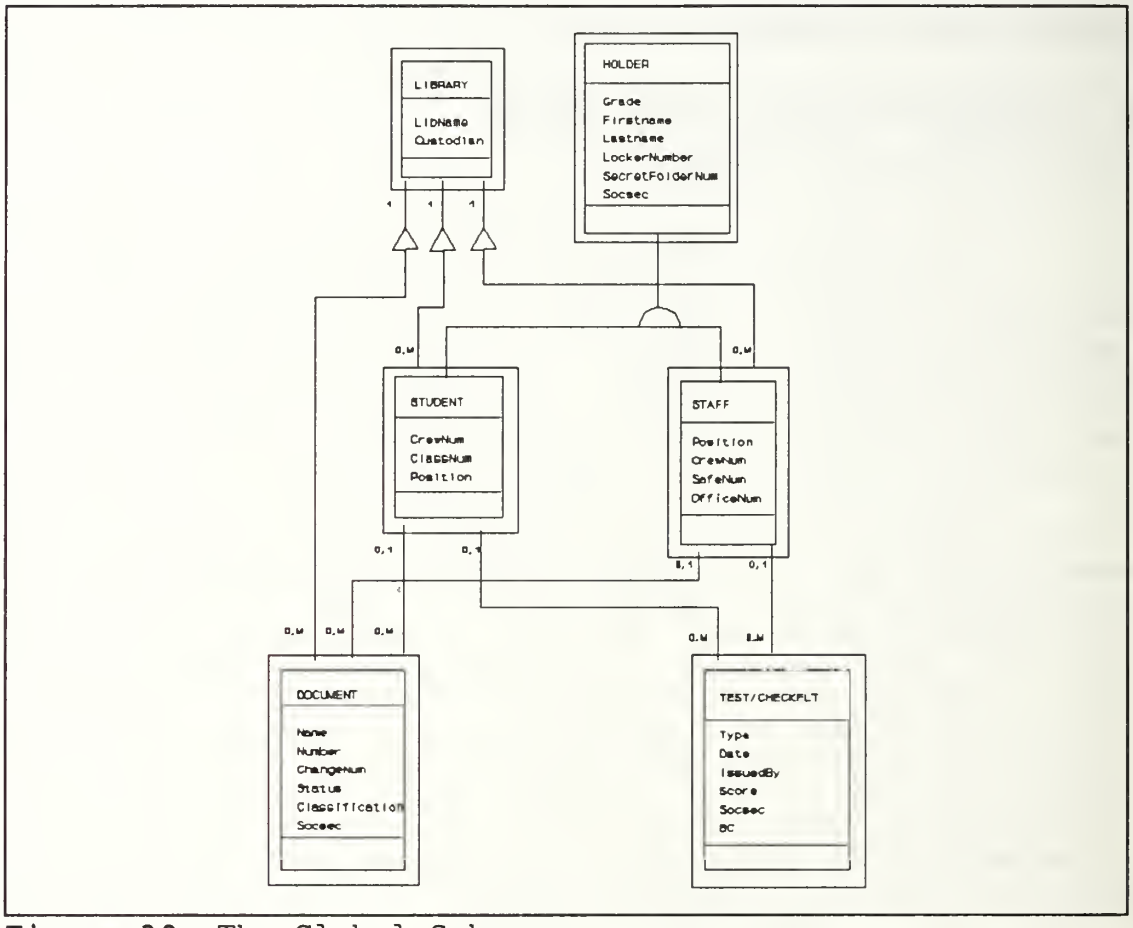

Figure 30 The Global Schema

C. THE GLOBAL CONTROLLER

The global controller was briefly mentioned earlier. This is an important component in the federated approach. It maintains the definition of the global schema and acts as a

coordinator and translator. When it receives a global query from a component database, it translates this query into an equivalent query on the global schema. This global schema query is decomposed and translated into subqueries that are sent to the corresponding local databases for processing. The results are collected and any corresponding data content conflicts are resolved. These results are then reformatted and sent back to the originating component database.

To illustrate this we look at the following query on the global schema from a relational user. The user wants a list of title and serial number of all publications checked out to a person whose social security number is 046-62-3436.

The global controller first transforms this into an equivalent query on the global schema. The information needed to make this change comes from a series of look-up tables.

The resultant query is then decomposed and translated into queries to the component databases. Again, extensive use of look-up tables enables this process. In our case we have two component queries, one being a query to the library database;

```
SELECT Publication, DocNumber
FROM Document, StudentHolder, StaffHolder
WHERE SS = 046623436.
```

The second is to the NATOPS databases;

```
SELECT PubName, PubNum
FROM Publication, StaffAircrew, StuAircrew
WHERE Socsec = "046623436".
```

The results of the component queries are collected and any corresponding data content conflicts are resolved by the global controller. The results are then reformatted and sent back to the requesting site.

All the conflicts and solutions are captured for use by the global controller. Additionally, the global controller maintains the definition of the global schema and acts as a coordinator and translator. At run-time it attempts to resolve the following data conflicts.

1. Data Inconsistencies Conflict Resolutions

The first conflict is caused by inconsistencies. These are generally due to failures in maintaining a database, such as failing to keep the database up to date. The global controller may not be able to resolve this conflict. One possible solution is to prioritize the component databases. If one database has a greater update rate than another, the assumption is that the data it contains is the most current. This can resolve some conflicts. However, it is not fool proof since it cannot correct for entry errors.

2. Different Representations for the Same Data Conflict Resolutions

Our framework has three categories of this conflict: different expressions for the same data, different units for the same data, and different granularities for the same data. In a global query for an individual's grade, the local NATOPS

model returns 'LT' and the Library model returns '0-3'. This is an example of different expressions for the same data. They both correspond to the U.S. Navy rank of lieutenant. The global comptroller must have some means of determining this similarity. Again, a solution is a look-up table.

The model does not have a different unit data conflict. This conflict occurs when two models use different units for similar numeric data. To illustrate this conflict, we examine two local databases that have an attribute for flight time. Querying one database yields the time in minutes, and the other yields the time in hours. One solution for translating the times is for the global controller to use a conversion formula.

As in the previous case this model does not have different granularity data conflict. These conflicts occur when two models use values from the domain of different cardinalities for the same data. To illustrate this, the NATOPS object test/checkflt has an attribute score. The data entered here is on a scale of 0.0 to 4.0. If another component database had a similar attribute where data is entered on a scale of 1 to 100. The global controller would need a mechanism to translate between the two. Again, possible solutions are look-up tables or conversion formulas.

D. CONCLUSION

This chapter proposed a method of resolving the identified conflicts between two component databases. To build a global schema that encompasses more databases, each successive database would be added in similar fashion to the resultant global schema of the previous component databases. In essence, building a global database with multiple component databases is an iterative process. The principles remain the same. However, the overall complexity increases. As this happens the importance of the global controller is magnified.

VIII. SUMMARY AND CONCLUSIONS

The framework developed in this thesis provides a comprehensive enumeration and classification of schema and data conflicts among component databases in an object-oriented database model. The schema conflicts are broadly classified by the level at which they could occur. These levels are: object level conflicts, attribute level conflicts, and object-attribute level conflicts. The data conflicts are classified as inconsistencies, and different representations for the same data. The following is a summary of these conflicts.

A. SUMMARY OF SCHEMA CONFLICTS

1. Object Level Conflict Summary

Object level conflicts occur when the heterogeneous databases use different representation for similar objects or abstractions. Object level conflicts are decomposed into object name conflicts and object structure conflicts.

2. Attribute Level Conflict Summary

Attribute level conflicts occur when the heterogeneous databases use different delineations to represent similar attributes. These conflicts are decomposed into attribute name conflicts, attribute constraint conflicts, and attribute structure conflicts.

3. Object-Attribute Level Conflict Summary

Object-attribute level conflicts occur when the same information is represented by one or more attributes in one model and as an object in another model.

4. Method Conflict Summary

The types of method conflicts are dependant on the strategy chosen for defining methods in the global model. One strategy would extend methods at the local schema level to apply to the global schema. In this situation, methods have to be rewritten or mapped into the final global schema that results from resolving all other conflicts

B. SUMMARY OF DATA CONFLICTS

1. Inconsistent Data Conflict Summary

Inconsistent data is generally due to data entry errors or failures in maintaining a database. Failures in maintaining a database usually manifest themselves in failing to keep the database up to date and failures to enforce semantic integrity rules.

2. Different Representations for the Same Data Conflict Summary

The three aspects of data that lead to different representation are different expressions for the same data (e.g., U.S. Navy, USN.), different units (e.g., inches, feet), and different granularity (e.g., a scale from 1 to 4 and a scale of 1 to 10).

C. APPLICATIONS

The Department of Defence is in the process of evaluating military information systems in regards to the corporate information management initiative (CIM). Many of the initial problems identified deal with redundant information systems. There is a need to access multiple independent information systems and to use the contained information for a strategic advantage at the department of defence level.

One solution is to consolidate these systems along lines of functionality, and rebuild them from scratch. The goal would be to reduce redundancies and foster interoperability between the remaining systems. This may not be feasible in every situation. An alternate solution is to organize existing systems along the lines of functionality, and then homogenize them so that they can share data. This is where resolving the heterogeneity conflicts becomes important.

D. FUTURE RESEARCH

Applying the framework and proposed solutions to build a global schema from a number of related component heterogeneous databases is the logical next step. Additional research is needed in the following areas.

1. Prototype Construction

The conflict framework and proposed solutions could be the basis of a prototype for building an information systems

homogenizing layer. The development of a workable prototype could impact how the Department of Defence proceeds with it's information consolidation efforts.

2. Development of Tools Based on Framework

The framework could be the foundation for a set of workable conflict identification tools. These tools could automate the identification and resolution of semantic and data conflicts found in similar databases prior to attempted integration. With the conflicts identified, the integration process should be significantly shortened.

3. Construct Artificial Intelligence (AI) Techniques to Resolve Semantic Issues

The conflict framework could be the foundation for an AI system that could automate the integration process between numerous component databases.

LIST OF REFERENCES

Bertino, E., and others, "An Object-Oriented Approach to the Interconnection of Heterogeneous Databases," In Position Papers of 1989 Workshop on Heterogeneous Databases, Dec 1989.

Brown, A., *Object-Oriented Databases: Applications in Software Engineering*, McGraw-Hill, 1991.

Bulman, D. M., and Bulman, E. K., "Objects,Entities,Things, and Knowledge (Object-oriented Programming Term Definitions)," *Computer Language*, vol 9, no 2, pp. 44-48, Jan 1992.

Coad, P., and Yourdon, E., *Object-Oriented Analysis*, 2d ed., Prentice-Hall, 1991.

Collet, C., Huhns, M. N., and Shen, W., "Resource Integration Using a Large Knowledge Base Carnot," *IEEE Computer*, pp. 55-62, Dec 1991.

Department of Defence Appropriations for 1991, *Automatic Data Processing Programs-Overview*, pp. 1-59, Government Printing Office, 1990.

Gillenson, M. L., *Database Step-by-Step*, John Wiley and Sons, 1990.

Heimbigner, D., and McLeod, D., "A Federated Architecture for Information Management," *ACM Transactions Office Information Systems*, vol 3, no 3, pp. 46-71, July 1985.

Kim, W.,and Seo, J., "Classifying Schematic and Data Heterogeneity in Multidatabase Systems," *IEEE Computer*, vol 24, no 12, pp. 12-18, Dec 1991.

Kroenke, D. M., and Dolan, K. A., *Databases Processing Fundamentals, Design, Implementation*, 3d ed, Macmillian Publishing Company, 1988.

Litwin, W., and Abdellatif, A., "Multidatabase Interoperability," *IEEE Computer*, pp. 10-18, Dec 1986.

Nolan, R. L., "Managing the Crises in Data Processing," *Harvard Business Review*, pp. 114-126, March-April 1979.

Notkin, D., and others, "Heterogeneous Computing Environments: Report on the ACM Sigops Workshop on Accommodating

Heterogeneity," *Communications of the ACM*, vol 30, no 2, pp. 24-32, Feb 1987.

Parsaye, R., and others., *Intelligent Databases*, Addison, 1989.

Sheth, A. P., and Larson, J. A., "Federated Database Systems for Managing Distributed Heterogeneous, and Autonomous Databases," *ACM Computing Surveys*, vol 22, no 3, pp. 183-236, Sep 1990.

Shlaer, S., and Mellor, S. J., *Object-Oriented Systems Analysis: Modeling the World in Data*, Prentice-Hall, 1988.

BIBLIOGRAPHY

Ahmed, R., and others, "The Pegasus Heterogeneous Multidatabase System," *IEEE Computer*, Dec 1991.

Edelstein, H. A., "Database World Targets Next-Generation Problems: As Distributed Applications Grow More Complex, Federated DBMSs, Object-Oriented Techniques Offer Solutions," *Software Magazine*, vol 11, no 6, pp. 79-85, May 1991.

Gupta,A., *Integration of Information Systems: Bridging Heterogeneous Databases*, IEEE Press, 1989.

Rafii, A., and others, "Integration Strategies in Pegasus Object Oriented Multidatabase Systems," paper from Hewlett-Packard Laboratories, 1991.

Wilkinson, K., Lyngbaek, P., and Hasan, W., "The Iris Architecture and Implementation," *IEEE Transactions on Knowledge and Data Engineering*, vol 2, no 1, Mar 1990.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Professor Magdi Kamel, Code AS/KA Naval Postgraduate School Monterey, California 93943-5002	9
4.	Professor Myung W. Suh, Code AS/SU Naval Postgraduate School Monterey, California 93943-5002	1
5.	Lieutenant Michael T. Bourque USS America (CV-66) FPO AE 09531-2790	1

Thesis
B73546 Bourque
c.1 A framework for classifying and resolving semantic heterogeneity in object-oriented databases.